VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Aplikace pro vizualizaci bio-signálů

Visualization of Bio-Signals

2014 Bc. Jan Marek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jan Marek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Aplikace pro vizualizaci bio-signálů
Visualization of Bio-Signals

Zásady pro vypracování:

Hlavním cílem práce je navrhnout a implementovat aplikaci pro vizualizaci bio-signálů, primárně EEG signálů. Aplikace musí umožňovat základní práci s daty typu zvětšení frekvenčního pásma, posuny dat, vizualizaci více kanálů a synchronní manipulaci s nimi, zvýrazňování, exporty a importy dat předem definovaných formátů. Předpokládá se rovněž návrh a implementace efektivní datové struktury pro práci s bio-signály. Požadovaný programovací jazyk a framework: C++, Qt.

1. Přehled existujících knihoven pro vizualizaci 2D grafů, zmapování jejich funkcí a případných možností využití.
2. Návrh a implementace vhodných datových struktur pro aplikaci.
3. Návrh a implementace aplikace pro vizualizaci bio-signálů s požadovanou funkčností.
4. Provedení výkonnostních testů aplikaci a jejich vyhodnocení.

Seznam doporučené odborné literatury:

[1] John Kessenich: The OpenGL Shading Language, http://www.opengl.org/documentation/glsl/, 2011
[2] Mark Segal and Kurt Akeley: The OpenGL Graphics System: A Specification,
2011
[3] Saeid Sanei, J. A. Chambers: EEG Signal Processing, Wiley-Interscience; 1 edition (September 11, 2007), ISBN-13: 978-0470025819

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Petr Gajdoš, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 2. května 2014 .....

Poděkování

Tato práce byla vypracována s podporou projektu Bio-inspirované metody: věda, vzdělávání a transfer znalostí, reg. č. CZ.1.07/2.3.00/20.0073 podpořeného Operačním programem Vzdělávání pro konkurenceschopnost, financovaného ze strukturálních fondů EU a státního rozpočtu ČR. Rád bych poděkoval svému vedoucímu práce Ing. Petru Gajdošovi Ph.D. za jeho ochotu, čas, trpělivost a cenné rady při tvorbě práce.

Abstrakt

Tato práce se zabývá možností zobrazování EEG signálů a práce s nimi. Hlavním cílem bylo navrhnout a implementovat aplikaci, která uživateli pohodlně umožňuje práci s těmito signály. Jde zejména o zobrazování dat ze senzorů headsetu, vyznačování důležitých úseků dat, jejich ukládání a načítání. V práci je uvedena teorie z oblasti elektroencefalografie a zpracování signálů (Fourierova a vlnková transformace, spektrogram, filtrace). Druhá část práce popisuje vytvořenou aplikaci a její ovládání.

Klíčová slova: EEG, C++, Qt, Vlnková transformace, Fourierova transformace

Abstract

This thesis deals with displaying and processing EEG signals. The main purpose was to suggest and implement application which offers comfortable work with these signals. Particularly it represents displaying data from headset sensors, marking significant parts of data sets, saving and loading data. Theory about electroencephalography, signal processing (Fourier and wavelet transform, spectrogram, filters) is introduced in the thesis. The second part of the thesis describes created application and its control.

Keywords: EEG, C++, Qt, Wavelet transform, Fourier transform

Seznam použitých zkratek a symbolů

EEG	–	Elektroencefalografie
NREM	–	Non-rapid eye movement – spánek s pomalými vlnami
AgCl	–	Chlorid stříbrný
Cl	–	Chlor
A/D převodník	–	Analogově digitální převodník, značeno také jako ADC
SVG	–	Scalable Vector Graphics – formát souboru popisující vektorovou grafiku
DOM	–	Document Object Model – objektová reprezentace XML souboru
USB	–	Universal Serial Bus – sběrnice, pomocí které lze připojit periferie k počítači
FFT	–	Fast Fourier transform – algoritmus výpočtu Fourierovy transformace

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Biosignály lidského těla jsou generovány nervovými a svalovými buňkami. Pomocí těchto signálů lze sledovat informace o nějakém biologickém systému – lidském organizmu. Elektrické biosignály můžeme získat snímáním povrchu orgánu nebo organizmu pomocí plošných elektrod. Získáme tak časový průběh daného biosignálu. Těchto signálů, jakožto aktivních elektrických projevů v lidském těle, může být více druhů podle toho, který orgán snímáme. Uvedeme několik příkladů:

- EKG – biosignál srdce.
- EEG – biosignál mozku.
- EMG – biosignál kosterních svalů.
- EGG – biosignál žaludku.
- EOG – biosignál okohybných svalů.

V této práci se budeme zajímat zejména o EEG signály. Cílem práce bylo vytvořit program pro vizualizaci těchto signálů a práci s nimi. Obecně je v programu možné vizualizovat libovolný signál, ale některé konkrétní části (např. filtrování vln EEG signálu) mají smysl pouze pro EEG signály. Největší důraz byl kladen na uživatelské rozhraní programu. Bylo třeba navrhnout takové rozhraní, které by umožňovalo co nejjednodušší a nejefektivnější práci se zachycenými signály. EEG záznamy bývají v praxi velmi rozsáhlé (typicky několikahodinový záznam). V takovém záznamu je obsaženo velké množství informací a je nutné se v nich snadno a rychle orientovat. Vzhledem k tomuto požadavku bylo nutné navrhnout nějakou metodu pro zvýrazňování a poznačování specifických částí (úseků v grafu, které jsou význačné). Takovýchto úseků však může být v záznamu hodně, je tedy nutné pro jejich výběr zvolit co nejvíce efektivní řešení. Dále bylo nutné implementovat metody pro zpracování signálu, které jsou nutné k analýze a vhodně výstupy z těchto metod zobrazit uživateli, teoretický základ k těmto metodám bude popsán v kapitole 2.4.

V práci je uvedena obecná teorie k EEG signálům a možnostem jejich zpracování, v další kapitole je rozebrána implementace samotného programu a popis ovládání. Kapitola testování se zabývá výkonem aplikace a jejími limity.

2 Elektroencefalografie

Jde o neurofyziologickou vyšetřovací metodu. Z vyšetřovacích metod mozkové činnosti patří k jedné z vůbec nejstarších. Zakladatelem této metody je Hans Berger, jenž své první zkušenosti s EEG publikoval v roce 1929 [12]. Výhoda této metody spočívá v nízkých nákladech spojených s pořízením a zpracováním EEG záznamů, nevyžaduje také podání radioaktivních látek ani přítomnost silného magnetického pole. Toto vyšetření lze také opakovat bez významné zátěže pacienta. Mezi nevýhody lze zařadit závislost na subjektivním úsudku vyšetřujícího spolu s rizikem chybné interpretace výsledného popisu. Jde o neinvazivní metodu.

Při elektroencefalografii se snímá pomocí povrchových elektrod bioelektrická aktivita mozku pacienta. Takto získaný záznam se nazývá elektroencefalogram. Tyto elektrody mohou být umístěny na povrchu hlavy (EEG), být v přímém kontaktu s kůrou mozkovou (ECoG, elektrokortikogram), nebo lze použít hloubkové elektrody při operacích (SEEG, stereoencefalogram). Podrobněji si možnost zachytit takovýto záznam popíšeme v kapitole 2.3. Elektroencefalografie je diagnostickou metodou, která se hodí zejména pro vyšetřování epilepsií. V současnosti je tato metoda v mnoha aplikacích nahrazována moderními neurozobrazovacími metodami (například magnetická rezonance), v některých oborech má však stále nezastupitelné místo (indikace záchvatových onemocnění – již zmíněná epilepsie). Informace v této kapitole jsou čerpány zejména z publikací [2, 12, 13, 14, 15].

2.1 EEG signál

EEG signál je součet všech elektrických dějů snímaných elektrodou [1]. Jde o záznam zesílených napětí mezi elektrodami v čase. Takovýto záznam lze zobrazit v grafu, kdy na ose X zaznamenáváme čas a na ose Y měnící se napětí mezi elektrodami. Takovýto záznam se v klinické praxi získává z mnoha elektrodových párů, podrobněji uvedeme v kapitole 2.3.

V šedé kůře mozkové můžeme rozpoznat tyto druhy potenciálů:

1. Jednotkové – vznikají na těle neuronu nebo axonu. Trvají krátkou dobu, řádově milisekundy.
2. Synaptické – trvají 15–40 ms.
3. Dendritické – mají komplexnější tvar a trvají až 0,1 s.

Dendritické a synaptické potenciály v povrchních vrstvách kortexu se nejvíce podílejí na vznik EEG signálu. EEG vzniká součinností neuronů thalamu a kortexu, přičemž thalamus plní funkci generátoru rytmů. Sumace elektrických potenciálových změn se v kůře děje hlavně na velkých, vertikálně orientovaných pyramidových buňkách, jejichž dendrity prostupují mnoha vrstvami mozkové kůry [13].

Dále uvedeme důležité parametry EEG signálu:

- Amplituda – přesné měření nemá praktický význam, amplituda závisí na mnoha faktorech (například vzdálenosti elektrod).

- Tvar – vztahuje se k morfologii vzorců. Může být pravidelný, nebo nepravidelný.
- Distribuce – značí elektrodu, nebo elektrody ve kterých je vzorek nejlépe zachycován. Pokud je maximum pouze v jedné elektrodě, jde o fokální aktivitu.
- Symetrie – vztahuje se k amplitudě. Určuje konstantnost, se kterou je jeden vzorec nižší, nebo vyšší než druhý.
- Synchronie – určuje vzájemný časový vztah jednotlivých vln a vzorců (výskyt ve stejné fázi a stejném čase). Rozlišujeme tyto druhy: *simultánní* (současný výskyt vzorců, nejsou ve fázi), *bilaterální* vzorce zachycované ve stranově korespondujících elektrodách, *asynchronní* (vzorce bez konstantního časového vztahu objevující se nad oběma polovinami hlavy) a *nezávislé*.
- Rytmicita – vlny se mohou opakovat se stejnou frekvencí, těm říkáme rytmické, nebo se opakují nepravidelně, což jsou arytmické.
- Periodicita – popisuje opakující se transienty.
- Perzistence – značí se indexem, který udává, procentuální množství výskytu vzorce v grafu. Jde o frekvenci, se kterou se v delším úseku grafu vyskytuje určitý vzorec.
- Reaktivita – změna, kterou lze navodit vnějšími stimuly.

2.2 Vlny v EEG signálu

Jako vlny se v EEG signálu označují všechny potenciálové změny (výchylky). Sledu takovýchto vln říkáme aktivita. Vlny můžeme dělit na pomalé, mají frekvenci nižší než 8 Hz a na rychlé, ty jsou s frekvencí vyšší. Za přirozený elektroencefalografický záznam můžeme považovat takový, který má u všech složek frekvenci přibližně do 80 Hz a amplitudu do 300 µV. Hodnota amplitudy se měří jako rozdíl minimální a maximální hodnoty. Podle výskytu typických rytmů lze rozdělit EEG signál do jednotlivých frekvenčních pásem. Jejich pojmenování vychází z historického hlediska. V daných frekvenčních pásmech lze pozorovat, že elektrická aktivita mozku je rytmická. Tyto rytmy se liší amplitudou, dále lokalizací na povrchu lebky a vztahem k různým fyziologickým stavům (mentální aktivita, spánek). V následující tabulce se přehledně popíšeme jednotlivé vlny a jejich frekvence.

Název pásma	**Frekvence [Hz]**	**Stav**
Delta	0,5–4	Hluboký spánek
Theta	4–8	Ospalost, usínání
Alfa	8–12	Bdělý stav se zavřenýma očima
Beta	12–30	Bdělí stav
Gama	nad 30	Extrémní zátěžové situace

Tabulka 2.1: Typy vln v EEG signálu.

Následně uvedeme krátký popis jednotlivých vln:

- *Delta* – Tyto vlny se běžně vyskytují ve spánku a to na frekvencích 0,5 a 1–4 Hz s amplitudou 20–200 µV. Jde o nejpomalejší vlny s nejvyšší amplitudou. U kojenců jsou ovšem běžné v bdělém stavu, zatímco u dospělých je v bdělém stavu označujeme jako patologické. Tyto vlny se vyskytují hlavně v hlubokém spánku (NREM), dále pak v transu, či hypnóze.
- *Theta* – Pásmo vln s frekvenčním rozsahem 4–8 Hz a amplitudou do 50 µV. Objevuje se při usínání a poté přechází v delta aktivitu. Toto pásmo je také zřídka kdy rytmické. U starších dětí a dospělých se objevuje právě při usínání nebo meditaci, zatímco u malých dětí se vyskytuje normálně v bdělém stavu. Tyto vlny se vyskytují v temporální a parientální oblasti. Pokud je jejich amplituda nejméně dvakrát vyšší než u alfa vln, indikují patologický stav. Tyto vlny se často pojí s fantazií, obrazovou představivostí, sny a živými vzpomínkami.
- *Alfa* – Frekvenční pásmo těchto vln se pohybuje v rozmezí 8–12 Hz s amplitudou dosahující maximálně 50 µV. Jde o první projev elektrického napětí generovaný lidským mozkem, který se podařilo zaregistrovat. Zaznamenal a pojmenoval ho Hans Berger. Tato aktivita je na obou stranách zadní části hlavy a má vyšší amplitudu na dominantní straně. Nejpatrnější je při zavřených očích a relaxaci. Tyto vlny jsou tvořeny aktivitou optického analyzátoru (lidé, kteří se narodí slepí, tuto aktivitu nemají vytvořenou).
- *Beta* – Vyskytují se ve frekvenčním pásmu od 12 Hz do 30 Hz s velikostí amplitudy okolo 30 µV. Vyskytují se na obou stranách ve frontální oblasti. Objevují se při duševní činnosti a jsou spojené s motorikou. Obecně bývají utlumeny při pohybu. Nízká amplituda a proměnná frekvence v tomto rytmu je spojována s myšlením a koncentrací. Naopak dlouhodobá rytmická aktivita je spojována s chorobami nebo užitím léků.
- *Gama* – Jde o pásmo s frekvencemi vyššími než 30 Hz a amplitudou v rozmezí 3–5 µV. Tyto vlny se objevují při vazbě různých populací neuronů za účelem vykonání kognitivní či motorické funkce.

Někdy mohou být dále rozlišovány:

- *Mí* – Jsou to vlny ve frekvenčním rozsahu 8–10 Hz a amplitudou v rozsahu 20–50 µV. Jde tedy o pásmo ve frekvenční oblasti *alfa* vln, které je ovšem utlumeno při pohybu.
- *Sigma* – Vlny vyskytující se v pásmu okolo 14 Hz a amplitudou 40 µV. Tyto vlny se vyskytují v třetím spánkovém stádiu.

Obrázek 2.1: Ukázka jednotlivých vln v EEG signálu [1].

Můžeme tedy říci, že normální vlny jsou v pásmech alfa a beta. Vlny, které mají nižší frekvence (tzn. nižší než 8 Hz) a vyskytují se častěji, jsou považovány za patologické. Ve spánku se však vyskytují fyziologicky. Ojediněle se mohou v EEG signálu vyskytovat vlnové tvary – garafoelementy. Mohou to být artefakty dvojího druhu:

1. Technické – z elektrod, přístroje, nebo vnějšího rušení.
2. Biologické – mrkání a pohyb očí, polykání, kašel, atd.

2.3 Zachytávání signálu

Pro zachycování signálů v biologických systémech jsou důležité proudy iontů. Je třeba uzavřeného obvodu, pro konstantní elektrický proud. Průchod proudu je zamezen, pokud je obvod otevřený a dochází tak k hromadění náboje na jeho koncích. Můžeme tak mluvit o velmi nedokonalém kondenzátoru. Při zachytávání EEG signálu lze hovořit o náhradním elektrickém obvodu složeném z rezistorů (tento charakter mají tělesné tkáně, rozhraní mezi skalpem a elektrodou, vlastní elektroda a vnitřní odpor EEG přístroje), kondenzátorů (takto se chová například mozkomíšní mok, lebka, nebo skalp) a induktorů.

Jednou z nejdůležitějších součástí EEG přístroje jsou elektrody, protože na jejich vlastnostech závisí kvalita snímaného signálu. Musí být vyrobeny z materiálu, který je dobře vodivý a zároveň nereaguje s biologickými elektrolyty. Elektrody musí být také nepolarizované, proto jsou vhodnými materiály vzácné kovy (například zlacené elektrody) nebo se mohou používat elektrody s vrstvou AgCl kombinované s roztoky, které snižují odpor vznikající mezi lebkou a elektrodou. Místo roztoků se ještě mohou využívat gely a pasty s volnými zápornými ionty Cl. Měla by být změřena elektronová impedance, ještě než začne samotné měření a to z důvodů, aby nedocházelo k nepřesnostem v měření. Hodnota by se měla pohybovat mezi 100–5000 Ω. Pokud je impedance nižší, dochází tak ke zkratu (bývá způsobeno vznikem vodivého můstku z potu, gelu nebo pasty). Naopak při vysoké impedanci bývá nedokonalý elektrický či mechanický kontakt. Impedanci tvoří tři základní složky a to odpor R, kapacitní reaktance X_C a induktivní reaktance X_L. Počítá se dle vztahu 2.1.

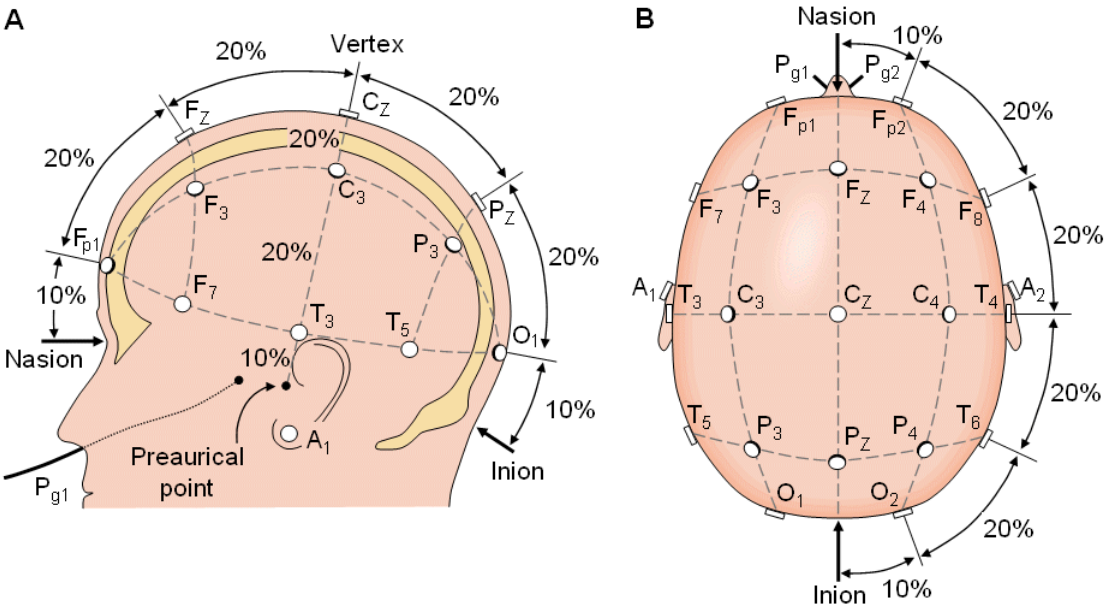
$$Z = \sqrt{R^2 + (X_C + X_L)^2} \tag{2.1}$$

Existuje mnoho typů elektrod využívaných při vyšetření. Přehledný seznam si uvedeme v následující tabulce:

Typy elektrod	Subtypy elektrod	Provedení
neinvazivní	1. skalpové	
semiinvazivní	2. semiinvazivní	subtemporální (sfenoidální), foramen ovale
invazivní	3. kortikografické	subdurální gridové, subdurální stripové
	4. hloubkové	

Tabulka 2.2: Typy elektrod využívaných pro měření EEG.

Elektrody se na povrch hlavy rozmísťují podle určitého schématu. Nejčastěji se elektrody umísťují dle systému 10–20. Název je odvozen podle toho, jak jsou jednotlivé elektrody rozmístěny. Podle tohoto systému se obvod hlavy rozdělí na úseky po 10% a 20%. Stejným způsobem se rozdělí hlava i ve zbylých dvou kolmých rovinách. Patrné to bude na následujícím obrázku:



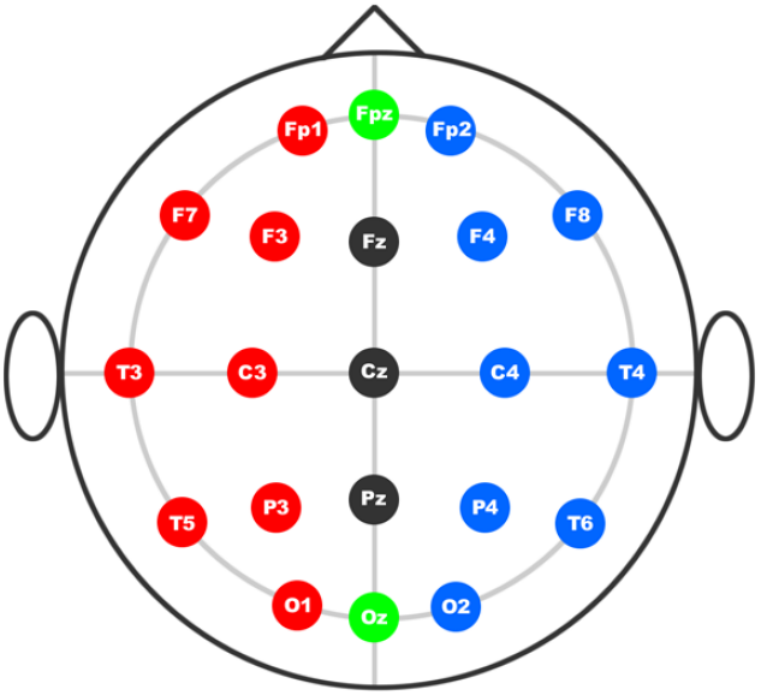
Obrázek 2.2: Rozložení elektrod podle schématu 10–20, A – boční pohled, B – pohled shora [10].

Dle tohoto systému je definováno 19 základních elektrod. Pro experimentální účely lze využít maximálně 128 elektrod. Toto omezení vyplývá z možného prostorového umístění na lebce. Při těchto vyšších počtech použitých elektrod se již používá systém 10–10, který dělí jednotlivé úseky na hlavě po 10%. Na následujícím obrázku ukážeme pozici EEG elektrod dle systému 10–20. Uvedeme také význam jednotlivých zkratek v názvu elektrod:

- C – centrální oblast,

- P – parientální (temenní lalok),
- F – frontální (čelní lalok),
- Fp – frontopolární (čelní lalok),
- O – okcipitální (týlní lalok),
- T – temporální oblast (spánková).

Číselné indexy označují laterální polohy, sudá čísla se vyskytují v pravé polovině a naopak lichá zase v levé polovině.



Obrázek 2.3: Pozice EEG elektrod podle schématu 10–20 [9].

Součástí každého digitálního EEG přístroje musí být A/D převodník. Jeho rozlišení by mělo být minimálně 8 bitů. Samotný EEG signál má velmi nízké hodnoty napětí (v řádu mikrovoltů), tudíž další nezbytnou součástí je zesilovač. Používají se diferenční zesilovače, pomocí kterých jsou zesilovány potenciály mezi dvěma elektrodami. Takovéto zesilovače musí mít zejména vysoký vstupní odpor (v řádu MΩ až desítek GΩ), který neovlivňuje měřený proces, zanedbatelný unikající proud, minimální vlastní šum a měnitelné nastavení zesílení (jednotlivé svody mají různý odpor, je tedy nutné dodržet měřítka jednotlivých kanálů).

V této práci bylo využito EEG přístroje od firmy Emotiv - Emotiv EPOC Neuroheadset [3], se kterým byla také aplikace testována. Aplikace je navržena pro práci s tímto konkrétním přístrojem a aplikačním rozhraním, které nabízí. Toto rozhraní bude podrobněji

popsáno v kapitole 3.1 a knihovny, které rozhraní zprostředkovávají pak v kapitole 3.3. Samotná vizualizační část aplikace není pevně vázána na konkrétní přístroj. Pokud tedy dostane přístup k vhodně formátovaným datům, může s nimi pracovat. Detailní popis návrhu aplikace bude podrobně popsán v kapitole 3. Přístroj se skládá ze samotného headsetu 2.4, USB vysílače, balení se šestnácti snímači, fyziologického roztoku a USB nabíječky.

Obrázek 2.4: Emotiv EPOC headset [4].

Kompletní specifikace tohoto přístroje jsou přehledně uvedeny v tabulce 2.3. Jde o výpis jednotlivých parametrů převzatý z oficiální specifikace [4].

Parametr	Hodnota
Počet kanálů	14 (plus referenční CMS/DRL a lokalizační P3/P4)
Názvy kanálů (dle standardu 10–20)	AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4
Vzorkovací metoda	Sekvenční vzorkování, jeden analogově digitální převodník
Vzorkovací frekvence	128 SPS (vzorků za vteřinu), 2048 Hz vnitřně
Rozlišení	14 bitů 1 LSB (nejméně významný bit) = 0,51 µV
Šířka pásma	0,2–45 Hz, digitální notch filtr (filtr, který v signálu potlačí jednu frekvenci) na 50 Hz a 60 Hz
Filtry	zabudovaný digitální 5řadý sinc filtr (ideální filtr typu dolní propust)
Dynamický rozsah	8400 µV (pp)
Spojovací mód	AC coupled
Spojení	proprietární bezdrátové, pásmo 2,4 GHz
Napájení	LiPoly
Výdrž na baterii	12 hodin
Měření impedance	kontrola kvality spojení v reálném čase pomocí patentovaného systému

Tabulka 2.3: Specifikace headsetu Emotiv EPOC.

2.4 Spektrální analýza

V této kapitole se budeme zabývat již samotným zpracováním zachyceného EEG signálu. Ukážeme si základní metody pro převod zachyceného signálu z časové oblasti do frekvenční. Využití takovéto frekvenční analýzy pro posuzování EEG signálu se objevovalo už ve čtyřicátých letech dvacátého století. Jednalo se však spíše o experimentální využití. Až s příchodem moderní výpočetní techniky nastal v této oblasti posun. Frekvenční analýza signálu je důležitá v mnoha případech. Jsou to například identifikace artefaktů, získání ucelené představy o převládajících aktivitách, topografickém mapování a jiné. Základní metody, které jsou implementovány v programu jsou Fourierova transformace 2.4.1, Vlnková transformace 2.4.2 a zobrazení spektrogramu 2.4.3.

2.4.1 Fourierova transformace

Fourierova transformace patří k základům zpracování jak analogových, tak i digitálních dat. Umožňuje transformovat signál z časové oblasti do frekvenční a obráceně. Provádí rozklad obecného signálu na jeho harmonické složky. Takto vzniklé spektrum je spojitá funkce frekvence. Zobrazuje signál jako funkci spojitého času a transformuje ji na funkci úhlové frekvence, obě proměnné mají obor hodnot všechny reálná čísla. Jde o integrální transformaci s exponenciálním jádrem $e^{-j\omega t}$, kde $\omega = 2\pi f$ je úhlová frekvence v rad/s a

$j = \sqrt{-1}$ je komplexní jednotka. Počítá se dle vztahu 2.2.

$$X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}dt \tag{2.2}$$

Zpětná Fourierova transformace pak dle vztahu 2.3.

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} X(\omega)e^{j\omega t}d\omega \tag{2.3}$$

Funkce $x(t)$ je původní funkce a výsledek transformace $X(\omega)$ je její obraz. Ještě uvedeme nutnou podmínku existence této transformace – transformovaná funkce musí být po částech spojitá s konečným počtem bodů nespojitosti a integrál funkce $|x(t)|$ musí být konečný [5].

Signály, které se zpracovávají na počítači, nemají obor hodnot všechny reálná čísla. Jsou již navzorkované, což zajišťuje A/D převodník, který je součástí EEG headsetu. Časová proměnná v takovémto signálu má obor hodnot ležící v celých číslech. U takto vzorkovaných signálů, pokud chceme zachovat možnost zpětné rekonstrukce na signál se spojitým časem, je nutno dodržet podmínku, aby nejvyšší frekvence nenulové složky signálu byla nejvýše rovna polovině vzorkovací frekvence. Této podmínce říkáme Shannon-Kotelnikův vzorkovací teorém. Pro tyto signály nelze použít obecnou Fourierovu transformaci jak jsme si uvedli výše. Využijeme proto modifikaci Fourierovy transformace pro diskrétní signály, která se jmenuje Diskrétní Fourierova transformace (dále v textu DFT). Délka navzorkovaného signálu bude vždy konečná. Takovýto záznam můžeme považovat za jednu periodu periodického signálu. DFT transformuje tuto periodu posloupnosti vzorků konečné délky na jednu periodu konečné posloupnosti z frekvenční oblasti. Výpočet se provádí dle vztahu 2.4.

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-j2\pi nk}{N}}, \qquad k = 0, 1, ..., N-1 \tag{2.4}$$

A zpětná DFT pak podle vztahu 2.5.

$$x_n = \sum_{k=0}^{N-1} X_k e^{\frac{j2\pi nk}{N}}, \qquad n = 0, 1, ..., N-1 \tag{2.5}$$

Signál x_n je původním signálem a výsledek výpočtu transformace X_k je obraz DFT. Původní signál má tedy pevně daných N hodnot, z kterých vypočteme opět N hodnot spektra. Hodnoty frekvencí tohoto spektra začínají od 0 Hz a jsou od sebe vzdáleny Δf. Tato vzdálenost je definována jako:

$$\Delta f = \frac{f_s}{N}, \tag{2.6}$$

kde f_s značí vzorkovací frekvenci a N celkový počet vzorků. Jak již bylo řečeno DFT je definována pro všechny frekvence odpovídající indexům od 0 až do $N-1$. Ovšem

frekvence na indexech $N/2$ až $(N-1)$ nejsou fyzikálně možné, protože by byl porušen vzorkovací teorém, jak bylo uvedeno. Můžeme říct, že jsou to záporně rotující frekvence se zápornou imaginární složkou periodického spektra se středem na vzorkovací frekvenci f_s. Těmto spektrům se říká dvoustranná. V praxi však pracujeme s reálnými signály, proto je možné počítat jen s jednostrannými spektry. U jednostranného spektra je přiřazena celá energie signálu jen do kladných frekvencí $0 - f_N$, kde f_N je Nyquistova frekvence počítaná dle vztahu:

$$f_N = \frac{f_s}{2} \tag{2.7}$$

Takovéto spektrum má dvojnásobné hodnoty oproti dvoustrannému spektru. Ještě poznamenejme, že zbývající hodnoty spektra na indexech $N/2$ až $(N-1)$, jsou komplexně sdružené k hodnotám na indexech 0 až $(N/2)-1)$[6]. Maximální frekvenci pak lze vypočítat dle následujícího vztahu:

$$f_{max} = \frac{f_s}{2} = \frac{1}{2}.\frac{N}{T}, \tag{2.8}$$

kde T značí interval měření všech N hodnot.

Klasický algoritmus výpočtu Fourierovy transformace potřebuje N^2 komplexních násobení a stejný počet komplexních sčítání. Takovýto výpočet má tedy příliš vysokou složitost, a proto byl vytvořen algoritmus *FFT – Fast Fourier Transform*. Tento algoritmus byl vytvořen Cooleyem a Tookeyem a je nazýván "algoritmem decimování v čase"neboli "algoritmem DIT". Tímto postupem se kvadratická složitost snižuje na lineárnělogaritmickou, tedy $(N/2)log_2(N)$. Je také důležité zmínit, že FFT algoritmus pracuje s délkou signálu $N = 2^m$, kde m je přirozené číslo. Pro představu uvedeme tabulku, ve které je zřetelné snížení složitosti [6].

N	**DFT operace**	**FFT operace**	**Účinnost**
256	65534	1024	64:1
512	262144	2304	144:1
1024	1048576	5120	205:1
2048	4144304	11264	372:1

Tabulka 2.4: Porovnání rychlosti algoritmu FFT [6].

2.4.2 Vlnková transformace

Vlnková transformace (Wavelet Transform – WT) patří mezi víceměřítkové transformace. Výhodou oproti Fourierově transformaci je, že můžeme získat informaci o přítomnosti určitých frekvencí a zároveň o jejich časovém výskytu. Fourierova transformace rozkládá signál do spektrální roviny pomocí periodických sinusových a kosinusových funkcí, naproti tomu bázi vlnkové transformace tvoří časově omezené funkce, které se nazývají vlnky. Každá vlnka osciluje pouze v okolí bodu svého momentálního výskytu, což je

hlavní výhoda této transformace oproti jiným metodám, protože díky tomu můžeme získat dobrou prostorovou lokalizaci. Tato transformace se hodí zejména pro neperiodické a nestacionární signály [7].

Jak již bylo řečeno, vlnková transformace patří mezi víceměřítkové. Takové transformace účinněji zpracovávají a analyzují data, než jednoměřítkové. Víceměřítkové zpracování znamená, že data jsou zpracovávána v různých měřítkových rovinách, které obsahují různě velké detaily a lze je zkoumat nezávisle na sobě. Výhoda je ta, že určité vlastnosti, či rysy signálu mohou být snáze detekovatelné v různých rovinách. Pro představu uvedeme příklad s dvourozměrným obrazem. Když se na takový obraz díváme z dálky, pak jej vnímáme jako celek, který je poté složen z menších celků. Tyto menší celky jsou z pohledu vlnkové transformace součástí vyšších měřítkových úrovní. A pokud se k obrazu přiblížíme, můžeme zkoumat detaily, které odpovídají vyšším prostorovým frekvencím, tedy z pohledu transformace jsou součástí nižších měřítkových úrovní [7].

Vlnková transformace pro spojité signály (Continous Wavelet Transform – CWT) je okenní operací. Jádro této operace se získá posunutím a roztažením bázové vlnky, která se nazývá mateřská a je speciální kauzální finitní funkcí $\psi(t)$ s nulovou střední hodnotou. S vlnkovými transformacemi můžeme velmi přesně lokalizovat i prudké změny signálu, protože mohou být nadefinovány dle charakteru analyzovaných dat. Což je výhoda oproti Fourierově transformaci, která využívá harmonických funkcí. Vlnková transformace je definována dle vztahu 2.9 a jejím výsledkem jsou vlnkové koeficienty $w(s,p)$.

$$w(s,p) = \int_{-\infty}^{\infty} f(t) \cdot \frac{1}{\sqrt{s}} \overline{\psi\left(\frac{t-p}{s}\right)} dt, \tag{2.9}$$

kde $f(t)$ je analyzovaný signál, ψ je vlnka, $\overline{\psi}$ je komplexně sdružená vlnka, s měřítko, p poloha umístění vlnky na časové ose $(s, p \in \mathbf{R}, s \neq 0)$, t je čas a člen $1/\sqrt{s}$ slouží k normalizaci energie vlnky při změnách měřítka [7].

Pokud máme symetrickou vlnkovou funkci, hledá se míra podobnosti (korelace) mezi vstupním signálem a vlnkou s měnícím se měřítkem, díky tomu můžeme extrahovat různě velké detaily ze zpracovávaných dat. U velikosti měřítka platí přímá úměra, tedy s menším měřítkem jsou extrahovány i menší detaily odpovídající vyšším kmitočtům. Poté můžeme odděleně zpracovat různě velké detaily, lze se tak vyhnout změně nebo poškození součástí signálu, které zpracovávat nechceme. Vlnku ψ nemůžeme volit úplně libovolně. Je to z důvodu zajištění invertibility transformace. Vlnka musí splňovat určité podmínky. Jsou jimi nulová střední hodnota a vhodný frekvenční obsah:

$$\int_{-\infty}^{\infty} \psi(t) dt = 0 \qquad a \qquad \int_{0}^{\infty} \frac{|\Psi(\omega)|^2}{\omega} d\omega < \infty, \tag{2.10}$$

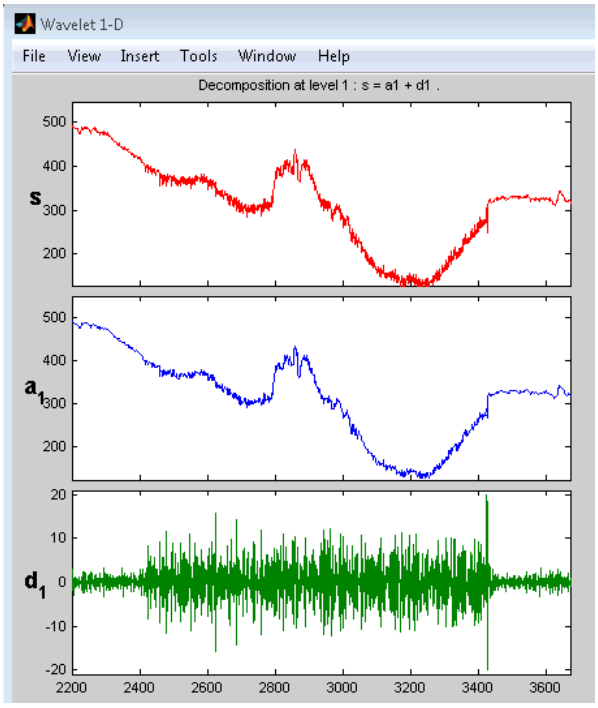
kde $\Psi(\omega)$ je Fourierův obraz $\psi(t) \in L^2(\mathbf{R})$ [7].

Stejně jako u Fourierovy transformace tak i u vlnkové nemůžeme použít tento obecný tvar pro práci se vzorkovanými signály. Opět je třeba použít modifikaci pro diskrétní signály – Diskrétní vlnková transformace (Discrete wavelet transform – DWT). Spojitá transformace nám dává velké množství dat, kterým říkáme vlnkové koeficienty. To je pro diskrétní signál nežádoucí, a proto se zde počítají pouze koeficienty, které odpovídají

měřítkům $s = 2^j$, kde $j = 0, 1, 2, ..., n$. Vhodnou závislostí parametrů s a p, s pomocí vyhovující vlnky ψ lze vytvořit ortonormální bázi: $s = 2^j, p = k \cdot 2^j, j, k \in Z$. Pak

$$\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}} \cdot \psi \left(\frac{n - k \cdot 2^j}{2^j} \right). \tag{2.11}$$

Dopřednou i zpětnou diskrétní vlnkovou transformaci poté můžeme spočítat podobným způsobem, jako u spojité varianty [7]. V tomto algoritmu diskrétní vlnkové transformace projde signál dvěma komplementárními transformacemi. První je dolní propust, která oddělí aproximaci prvního řádu a druhá horní propust zase detail prvního řádu. Touto filtrací získáme koeficienty aproximace 1. řádu kA1 a koeficienty detailu 1. řádu kD1. Opakováním tohoto postupu lze získat koeficienty aproximací a detailů vyšších řádu [16]. Příklad rozkladu signálu s nastavenou jednou dekompoziční úrovní a za použití vlnky db1 je na obrázku 2.5.

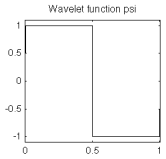


Obrázek 2.5: Ukázka rozkladu signálu [19].

I v případě vlnkové transformace byl vymyšlen algoritmus pro urychlení výpočtu. Tím nejznámějším je Matallův algoritmus (obecně Fast Wavelet Transform – FWT). Tento algoritmus spočívá ve filtraci signálu souborem kvadraturních zrcadlových filtrů typu dolní a horní propust [7]. Podrobný popis tohoto algoritmu je možné nalézt v článku časopisu Elektrorevue [8].

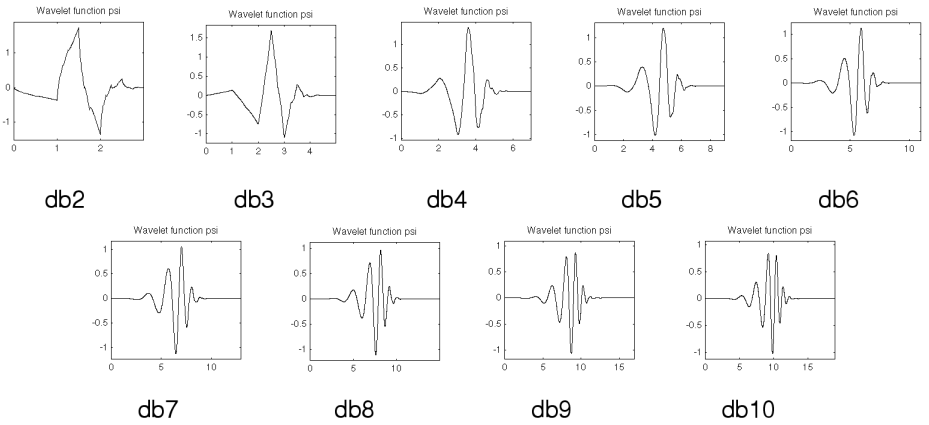
Vlnková transformace využívá řadu bázových vlnek. Jejich jména jsou většinou odvozena od jména tvůrce, nebo podle jejich tvaru. Uvedeme seznam několika vlnek, které budou k dispozici i v samotném programu.

- *Haar* – Tvarově jde o nejjednodušší vlnku. Je nespojitá, což je nevýhoda. Průběhem připomíná skokovou funkci a reprezentuje stejnou vlnku jako Daubichies db1.

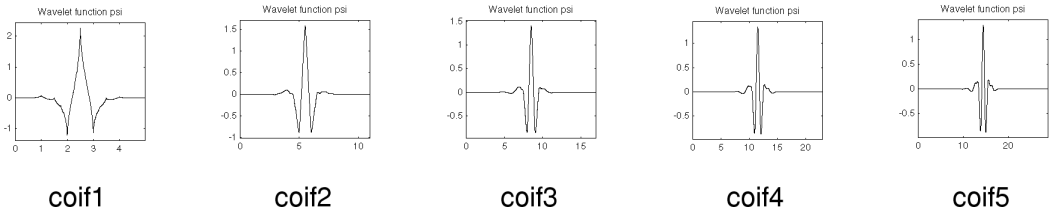


Obrázek 2.6: Haarova vlnka.

- *Daubechies a Coiflets* – Autorem těchto vlnek je Ingrid Daubechies. Jde o spojité a ortonormální vlnky. Značí se písmeny dbN pro vlnky Daubechies a coifN pro vlnky Coiflets, kde číslo N blíže určuje konkrétní vlnku.

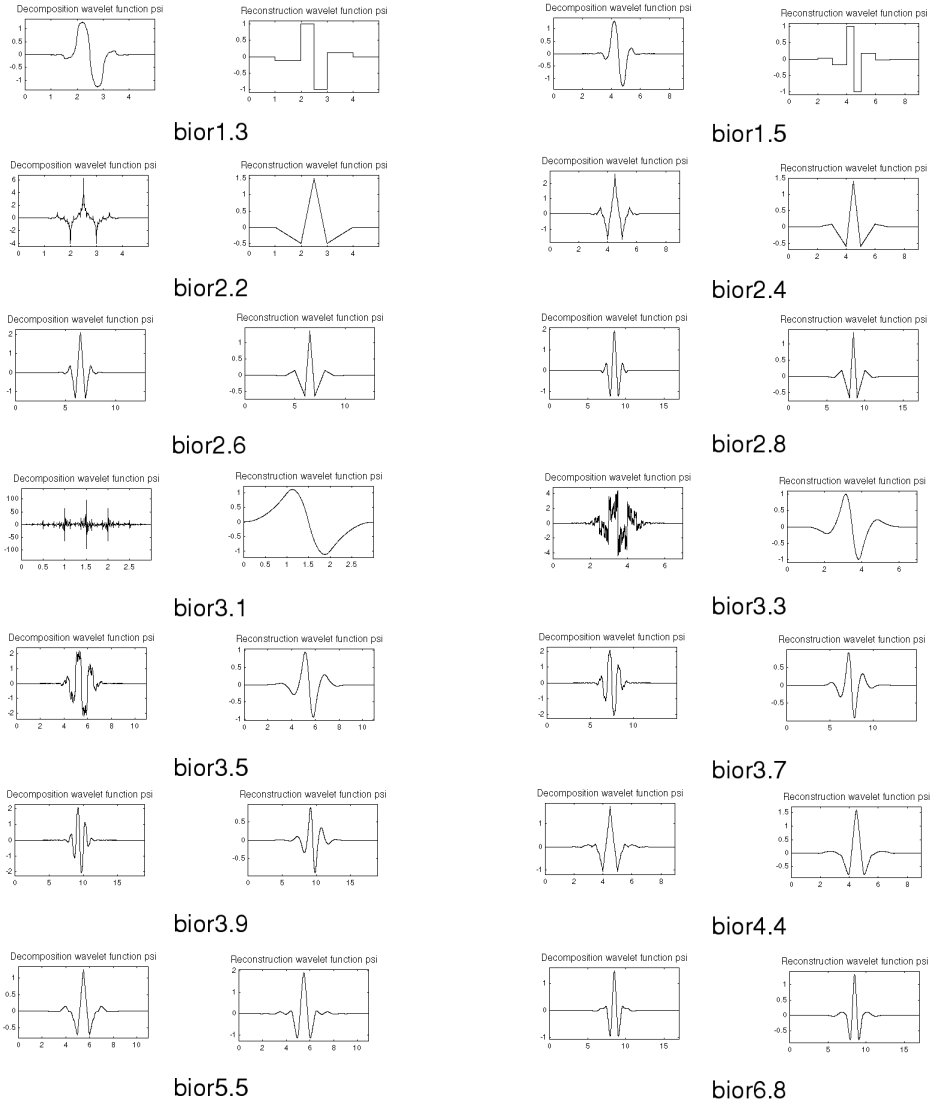


Obrázek 2.7: Vlnky skupiny Daubechies.



Obrázek 2.8: Vlnky skupiny Coiflets.

- *Biorthogonal* – Pomocí skupiny těchto vlnek můžeme odvodit některé důležité vlastnosti, které jsou potřeba při rekonstrukci signálu. Proto se používají dvě vlnky, kdy jedna slouží pro rozklad a druhá pro rekonstrukci signálu.



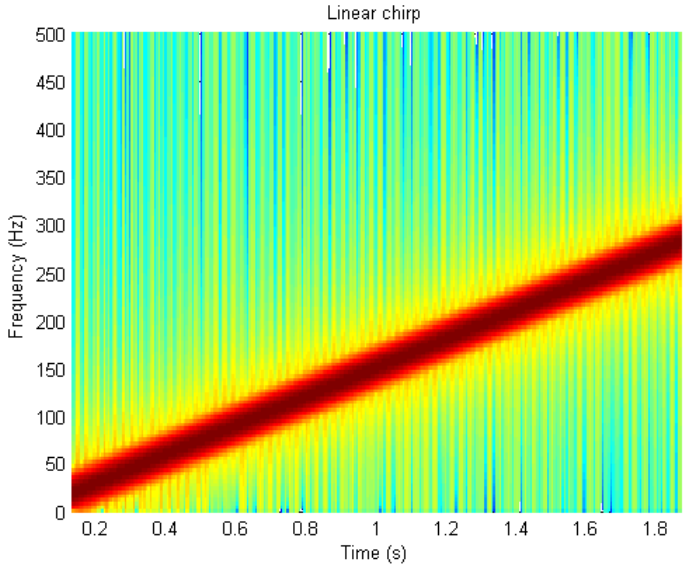
Obrázek 2.9: Vlnky skupiny Biorthogonal.

2.4.3 Spektrogram

Pomocí spektrogramu můžeme vyjádřit časově-frekvenční závislost. Jde o graf, kdy jedna osa popisuje čas (většinou vodorovná) a druhá frekvenci (většinou svislá). Zbarvení daného bodu potom určuje amplitudu. Ukázka spektrogramu je na obrázku 2.10.

Spektrogram může mít v grafu lineární, nebo logaritmické měřítko. V programu bude použito lineární, logaritmické měřítko se hodí pro zobrazení hudebních signálů. Pro tvorbu lineárního spektrogramu se využije Fourierova transformace. Ta se postupně aplikuje na jednotlivé časové intervaly analyzovaného signálu. Tato aplikace se nazývá

krátkodobá Fourierova transformace (Short-Time Fourier Transform – STFT). Takto transformované jednotlivé části signálu vyneseme do grafu jako svislé pruhy. Díky této metodě můžeme vytvářet spektrogram v reálném čase. Nevýhodou tohoto postupu je složitější tvorba logaritmických spektrogramů, což není v našem případě potřeba. Také zde platí všechna pravidla, jak jsme si uvedli u Fourierovy transformace, zejména výpočet maximální frekvence. Kvůli lepšímu rozlišení frekvencí a zabránění vzniků artefaktů, které mohou vznikat na přechodech mezi intervaly se Fourierova transformace nepočítá z po sobě následujících bloků dat, ale z postupně se překrývajících částí [17].



Obrázek 2.10: Ukázka spektrogramu [20].

2.5 Filtrace

Filtrování EEG signálu je důležité pro zvýraznění, nebo naopak potlačení určitých složek signálu. Tyto složky jsou patrné ve frekvenční oblasti signálu a mohou to být například jednotlivé EEG vlny. Pokud použijeme na signál nějaký filtr, změníme tím jeho amplitudu a časové vztahy mezi harmonickými komponentami. Nejčastěji se filtrování používá pro odstranění rušivých elementů. Použitím filtrů však může dojít i k potlačení užitečného signálu [17].

Filtry můžeme rozdělit podle typu impulsní odezvy na:

- *FIR* – filtr s konečnou impulsní charakteristikou (Finite Impulse Response). Jde obecně o nerekurzivní filtry (nezávisí tedy na předchozích výstupech) a jejich impulsní charakteristika má konečný počet hodnot. Realizační algoritmus je vyjádřen rovnicí 2.12.

$$y_n = \sum_{k=0}^{N=1} x_{n-k} h_k, \tag{2.12}$$

kde $x(n)$ je vstupní signál a $h(n)$ je impulsní charakteristika filtru [18].

- *IIR* – filtr s nekonečnou impulsní charakteristikou (Infinite Impulse Response). Výstup je závislý na předchozích výstupech. Jejich výhodou je menší náročnost na rozsah výpočtů než u FIR filtrů. Je ovšem složitější je navrhnout. Lze je vyjádřit pomocí vztahu 2.13.

$$y_n = \sum_{i=0}^{r} L_i x_{n-i} - \sum_{i=1}^{m} K_i y_{n-i}, \tag{2.13}$$

  kde L_i a K_i jsou systémové koeficienty v dopředných a zpětných vazbách, r značí počet zpoždění v nerekurzivní části a m počet zpoždění v rekurzivní části daného systému [18].

Dále můžeme dělit filtry dle frekvenčních složek, které propouští následovně:

- *Dolní propust* – vysokofrekvenční filtr (Low-pass filter). Tento filtr propouští pouze nízké frekvence. To znamená, že všechny frekvence vyšší než zadaná frekvence jsou ořezány.
- *Horní propust* – nízkofrekvenční filtr (High-pass filter). Tento filtr naopak propouští jenom vysoké frekvence. Opět podobně jako u dolní propusti jsou frekvence nižší než zadaná frekvence ořezány.
- *Pásmová propust* – vznikne kombinací obou předchozích filtrů (Band-pass filter). Tento filtr propouští pouze frekvence ve zvoleném pásmu.

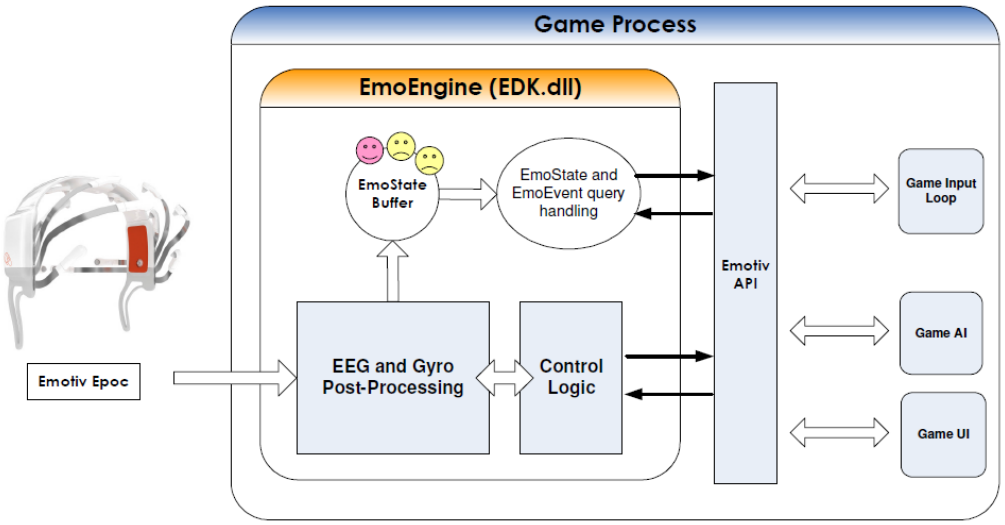
V programu jsou použity tyto filtry zejména pro filtrování konkrétních EEG vln.

3 Implementace programu EEG manager

V této kapitole bude popsána samotná implementace programu. Nejprve bude popsán obecný přístup a třídní návrh v podkapitolách 3.1 a 3.2 v podkapitole 3.3 výběr implementačních technologií a v dalších podkapitolách již detailní popis jednotlivých částí programu a jejich programová implementace. Budou také uvedeny příklady se zajímavými částmi kódu. Celý program se skládá ze dvou základních částí – jsou jimi *EEGDataCore* a *EEGManager*. Tento návrh umožňuje využití obou části nezávisle na sobě. Zatímco *EEGDataCore* se stará o samotné získávání dat z headsetu, jejich ukládání a zpracování, tak *EEGManager* má na starosti vytvoření uživatelského prostředí, zajištění interakce s uživatelem a vizualizaci dat. Pro aplikaci byl základem strukturální návrh (třídy pro ukládání dat a základní vizualizace) od doktora Petra Gajdoše.

3.1 Získání dat z headsetu

Získávání dat má na starost část programu *EEGDataCore*. Stará se o získávání dat z headsetu, obsluhu událostí a ukládání dat do bufferů. Pro tyto operace je využito *Emotiv SDK* (Software Development Kit), které obsahuje *Emotiv API* (Application Programming Interface) a hardwarový přijímač signálu z headsetu. Detailnější popis z čeho se skládá *Emotiv API* bude uveden v kapitole 3.3. Získávání dat je tedy uzpůsobeno pro testovaný headset od společnosti Emotiv, samotné ukládání do připravených struktur již však není závislé na konkrétním headsetu. Struktury pro ukládání dat, tak mohou sloužit i pro jiné zdroje dat. Popis jednotlivých tříd, které mají na starosti jak získávání, tak i ukládání dat bude uveden dále. Nyní pro ilustraci získávání dat uvedeme obrázek, na kterém je patrné jak se s daty zachází a jak k nim můžeme přistupovat přes již zmíněné aplikační rozhraní. Obrázek je převzat z uživatelského manuálu pro *Emotiv SDK*.



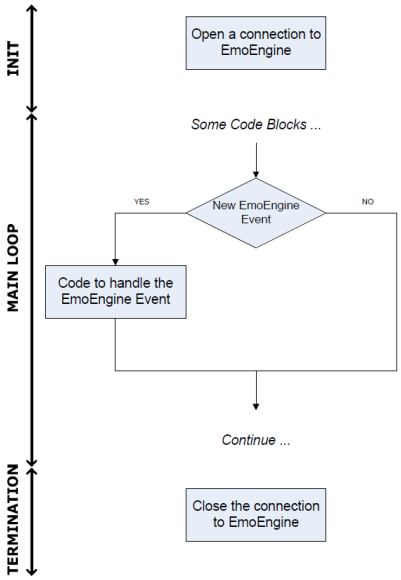
Obrázek 3.1: Komunikace s headsetem [11].

Na výše uvedeném obrázku je příklad interakce headsetu s videohrou. To je však naprosto obecné a místo videohry si můžeme představit náš program, který pomocí uživatelského rozhraní pracuje se samotným *EmoEnginem*.

Získávání a ukládání dat je v programu řešeno pomocí tří základních tříd. Nyní uvedeme jejich seznam s popisem k čemu slouží:

- *Headset* – Základní třída, sloužící pro práci s headsetem. Provádí nastavení parametrů headsetu, připojení se k němu a spuštění zachytávání dat. Dále nabízí možnost ukládání a načítání zachycených dat.
- *EEGBuffer* – Třída, která slouží pro práci se zachycenými daty. Poskytuje k nim přístup a uchovává další doplňující informace. K samotným zachyceným datům je třeba ještě uchovávat jisté metadata, ty například obsahují časovou složku od kdy a do kdy byla data zachytávána. Dále je zde nezbytné ukládat informace o uživatelem označených částech dat.
- *EEGDynArray* – Tato třída slouží již pro samotné uchovávání hodnot (v tomto případě číselných). Jde o třídu, která vytváří dynamické pole a poskytuje metody pro práci s ním.

Jak bylo uvedeno základní třída, která se stará o komunikaci s headsetem prostřednictvím jeho rozhraní je třída Headset. Třída je implementována jako vlákno, protože je nutné, aby objekt této třídy fungoval souběžně s ostatními částmi programu. Nejdůležitějšími operacemi je obsluha událostí vyvolaných headsetem. Přehledné schéma komunikace se samotným headsetem a obsluhou jeho událostí uvedeme v následujícím vývojovém diagramu:



Obrázek 3.2: Obsluha událostí [11].

Jak je z diagramu patrné, nejprve je nutné vytvořit spojení s headsetem, to je vytvořeno s uživatelem nastavenými parametry. Následuje blok kódu, který obsluhuje události headsetu. A nakonec je nutné uzavřít spojení. Nejdůležitější části je tedy obsluha událostí, tato část kódu pracuje v nekonečném cyklu, kdy se v určitém časovém rozestupu cyklicky kontroluje výskyt nové události a následně se zařídí případná obsluha. Aplikace obsluhuje tři základní typy událostí :

- *Hardware-related events* – Události vyvolané při připojení, nebo odpojení Emotiv headsetu k počítači (například *EE_UserAdded*).
- *New EmoState events* – Události vyvolané uživatelovým obličejem (například mrknutí, úsměv), změnou emočního stavu (například *EE_EmoStateUpdated*).
- *Suite-specific events* – Události týkající se trénování a konfigurace kognitivity a detekcí výrazů (například *EE_CognitiveEvent*).

Nyní můžeme uvést posloupnost kroků, kde si chronologicky znázorníme získání dat z headsetu a jejich uložení do připraveného dynamického pole, spolu se zaslaným oznámením o aktualizaci dat. Jak jsme uvedli výše, toto získávání dat pracuje v nekonečném cyklu (než jej uživatel přeruší). Tento cyklus můžeme vyjádřit body 3. – 5.

1. Připojení se k headsetu.
2. Obsluha události přidání uživatele. Headset musí pracovat s uživatelským ID, jinak není schopen přijímat data.
3. Kontrola, zda jsou připravena nová data k uložení.
4. Uložení zaslaného množství dat do dynamického pole. Každý kanál headsetu má své vlastní dynamické pole, kam postupně přidává data. Dále se ukládají metadata.
5. Zaslání signálu, který informuje o aktualizaci dat v bufferu.
6. Ukončení připojení k headsetu.

3.2 Zobrazení dat v grafech

Zobrazovaní dat v grafech a celkově veškeré grafické rozhraní má na starosti část *EEGManager*. Obsahuje několik tříd, které mají na starosti vytvoření hlavního okna aplikace, zajištění interakce s uživatelem (možnosti nastavení programu) a zobrazování samotných grafů. Podrobněji se jednotlivým grafickým částem programu budeme věnovat v dalších kapitolách. Nyní uvedeme základní přehled tříd, které mají na starosti samotné zobrazování.

- *SignalDockWidget* – Základní třída sloužící pro vytvoření hlavního okna aplikace a všech uživatelských prvků. Prvky této třídy jsou vytvořeny staticky dle předem navrženého uživatelského rozhraní. Tato třída se také stará o uživatelské interakce.

- *myScene* – Tato třída se stará o zobrazení obrázku s rozložením EEG senzorů. Také má na starosti interakci s uživatelem v podobě možnosti kliknutí na jednotlivé senzory. Tímto může uživatel vybraný senzor aktivovat, nebo deaktivovat. Tato třída vyžaduje jako zdroj obrázek ve formátu SVG.

- *SignalPlotAll*, *signalPlotSpectrogram*, *SignalPlotSpectrum* – tyto třídy mají na starosti samotné vykreslení grafů. Svým účelem jsou tedy podobné, ale jsou takto rozdělené, protože každá slouží k zobrazení trochu odlišného typu dat. Třída *SignalPlotSpectrum* slouží k zobrazení dat v reálném čase. Slouží k vykreslení nezpracovaných dat, ale i upravených a modifikovaných (například po Fourierově transformaci). Třída *signalPlotSpectrogram* slouží rovněž k zobrazení dat v reálném čase, ale jak již název napovídá, zobrazuje pouze spektrogram. Jde tedy o odlišné zobrazení dat, které bylo popsáno v kapitole 2.4.3. Třída *SignalPlotAll* poté slouží k zobrazení všech zachycených dat. Nejde tedy o zobrazení v reálném čase, ale až po ukončení odchytávání. Tato třída rovněž nabízí možnost interakce s uživatelem a to v podobě vybírání konkrétních úseků dat.

Hlavní okno aplikace je rozděleno do dvou vertikálních částí. Obě části mají poté tři panely. V pravé části je jeden panel pro výběr senzorů a další dva pro samotné zobrazení grafů. Panely v levé části pak slouží k nastavení programu. V pravé části jeden panel slouží pro hromadné zobrazení grafů k vybraným senzorům a druhý naopak pro detailní grafy k vybranému senzoru. Objekty příslušných tříd pro samotné zobrazení grafů (*SignalPlotAll*, *signalPlotSpectrogram*, *SignalPlotSpectrum*) jsou dynamicky vytvořeny a umístěny do připraveného grafického objektu, který má stromovou strukturu. Zobrazení dat v reálném čase je poté možné pouze v jednom panelu. Je to logické, neboť uživatel nemůže souběžně sledovat oba dva panely. Zdrojem dat pro všechny grafy jsou objekty tříd, které byly popsány v kapitole 3.1. Jak jsou jednotlivé grafické části prakticky implementovány a ukázky jednotlivých částí bude uvedeno v následujících kapitolách.

Nyní uvedeme chronologickou posloupnost, jak jsou od startu aplikace vytvořeny objekty pro zobrazení dat v grafech:

1. Je vytvořeno hlavní okno aplikace. V příslušných panelech je staticky vytvořena stromová struktura, při startu aplikace neobsahuje žádné objekty.

2. Po startu zachytávání dat jsou ve stromové struktuře vytvořeny objekty třídy *signalItemSpectrogram*, *SignalItemSpectrum*, nebo *SignalItemAll* v závislosti jaký druh dat mají zobrazovat. Jsou to jednotlivé položky stromové struktury.

3. Každá položka stromové struktury obsahuje objekt pro samotné zobrazení grafu. Jde o objekty tří tříd, které byly popsány výše.

4. Výsledné grafické objekty zobrazí data z připravených datových struktur.

3.3 Výběr technologií

V této kapitole budou představeny konkrétní technologie použité v aplikaci. Bude zmíněno, jak jsou implementovány důležité třídy a které knihovny se v programu využívají (zejména ty pro vykreslování grafů).

Celá aplikace je implementována v jazyce C++. Tento jazyk byl vybrán, protože *Emotiv API* poskytuje rozhraní pro jazyk *ANSI C*, lze jej tedy využít pro aplikace psané v jazyce C++. *Emotiv API* se skládá ze tří hlavičkových souborů (edk.h, EmoStateDLL.h, edkErrorCode.h) a ze dvou Windows DLL knihoven (edk.dll, edk_utils.dll). Jelikož je potřeba těchto knihoven, je aplikace psaná pro operační systém Windows.

Pro grafickou část aplikace byl zvolen grafický framework Qt. Ten se stará zejména o vykreslování uživatelských oken a prvků. Poskytuje však další užitečné struktury a třídy. Jednou z nich je zejména třída pro práci s vlákny. Jak bylo řečeno v kapitole 3.1 třída *Headset* je implementována jako vlákno, tato třída tedy dědí z Qt třídy – *QThread*. Qt využívá pro komunikaci mezi objekty signály a sloty. Tato komunikace je v aplikaci využita pro dva účely. Prvním je interakce uživatele s grafickým rozhraním a druhá pro oznamování aktualizace dat (například objekt třídy *Headset* oznamuje objektu třídy *SignalPlotSpectrum*, že má dostupná nová data, která jsou připravena k vykreslení do grafu). Qt má dva základní nástroje pro vytváření grafického rozhraní – jedním je možnost využít jazyka QML a druhým využití widgetů. V aplikaci je zvolena práce s widgety a to z důvodu kompatibility s knihovnou pro vykreslování grafů, která pracuje právě s widgety.

Dále bylo nutné vybrat knihovnu pro samotnou vizualizaci grafů. Dvě nejdostupnější varianty pro Qt jsou knihovna Qwt (Qt Widgets for Technical Applications) a QCustomPlot. Obě knihovny nabízí možnost vykreslení jednoduchých grafů, které jsou potřeba při zobrazování EEG signálů. V aplikaci je však umožněno vykreslování i spektrogramů, které v době implementace programu bylo v základu možné jen s knihovnou Qwt. Dále bylo nutné zajistit možnost uživatelského výběru konkrétních úseku v grafu, tedy bylo nutné, aby knihovna poskytovala mechanismus výběru konkrétního úseku z grafu co nejjednodušším způsobem (například tažením myši). V základu toto umožňuje pouze knihovna Qwt, a proto byla vybrána a použita v aplikaci. Všechny tři třídy pro samotné vykreslení grafů (jak bylo zmíněno výše v kapitole 3.2) dědí ze základní třídy *QwtPlot* (z knihovny Qwt). Každá taková třída pak má objekt třídy *SignalCurve*, což představuje křivku grafu. Samotná data k vykreslení jsou pak předána pomocí metody:

*setRawSamples(const double *xData, const double *yData, int size)*.

Metoda vyžaduje dva ukazatele na data pro osy X, Y a celočíselnou hodnotu, která značí kolik hodnot má být vykresleno.

Poslední důležitou součástí programu jsou dvě třídy, které se starají o zpracování dat pro spektrální analýzu. Jsou to třídy *dataSpectrum* a *dataTransform*. První z nich zajišťuje transformaci dat pro vykreslování spektrogramu a druhá pak výpočet Fourierovy transformace a vlnkové transformace. Třída *dataTransform* také zajišťuje filtrování signálu ve frekvenční oblasti. Tyto třídy provádí dvě základní transformace s daty – Fourierova a vlnková transformace. K těmto operacím využívají specializované knihovny. Knihovna pro výpočet diskrétní Fourierovy transformace je *FFTW* (Fast Fourier Transform in the West). Jde o knihovnu v jazyce C, která dokáže počítat dopřednou i zpětnou transformaci,

umí pracovat s reálnými a komplexními čísly. Knihovna pro výpočet vlnkové transformace byla zvolena *wavelet1d* (1D/2D Discrete Wavelet Transform Implementation in C++). Jak je z názvu patrné, knihovna je napsaná v jazyce C++ a sama využívá také knihovny *FFTW*. Nyní uvedeme krátkou ukázku využití obou knihoven. Knihovna *FFTW* využívá pro práci s daty speciální strukturu – *fftw_complex*, pro kterou se alokuje paměť funkcí *fftw_malloc* a poté uvolňuje funkcí *fftw_free*. Datová struktura je implicitně typu double[0] a uchovává komplexní čísla. Dále je potřeba vytvořit takzvaný *plan*. Jde o objekt, který obsahuje všechna potřebná data k transformaci. Tento *plan* je po vykonání transformace nutné smazat pomocí funkce *fftw_destroy_plan*. V programu se využívá varianta transformace reálných (jednorozměrných) dat. Následující kód znázorňuje takovouto transformaci, kdy na vstupu jsou data zachycená headsetem a na výstupu komplexní čísla po Fourierově transformaci.

```
double *in;
fftw_complex *out;
fftw_plan p;
out = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * (N/2 - 1));
p = fftw_plan_dft_r2c_1d (N, in, out, FFTW_MEASURE);
fftw_execute(p);
fftw_destroy_plan(p);
fftw_free (out);
```

Výpis 3.1: Ukázka Fourierovy transformace pomocí knihovny *FFTW*.

Vstupní pole *in* musí být alokované na velikost, která je rovna mocnině čísla 2. Označme tuto velikost jako N, výstupní pole *out* musí mít velikost nejméně $N/2 - 1$. Značka *FFTW_MEASURE* instruuje *FFTW*, aby před samotným výpočtem provedla měření a zvolila nejvhodnější způsob výpočtu transformace.

Funkce pro výpočet vlnkové transformace potřebuje několik parametrů. Předpis funkce si ukážeme na následujícím výpise:

```
void* dwt(vector<double> &sig, int J, string nm, vector<double> &dwt_output, vector<double>
    &flag, vector<int> &length )
```

Výpis 3.2: Funkce pro výpočet vlnkové transformace.

Zdrojová data funkce vyžaduje ve struktuře jazyka C++ a to *vector*, zde jde o proměnnou *sig*. *J* je parametr nastavující úroveň dekompozice, *nm* označuje jméno bázové vlnky, ve vektoru *dwt_output* jsou pak uloženy výsledky transformace, vektor *flag* je dvourozměrný, kdy první hodnota označuje zda je signál sudý (0) nebo lichý (1) a druhá pak úroveň dekompozice, poslední vektor *length* obsahuje postupně délky jednotlivých výsledných vektorů.

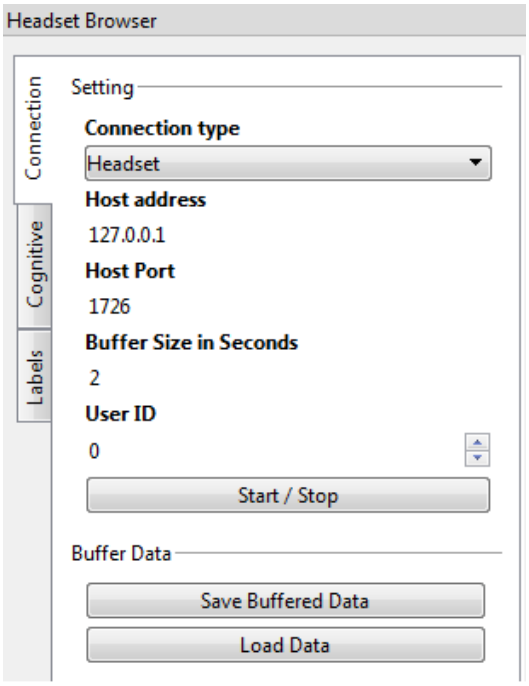
3.4 Nastavení programu

V následujících třech podkapitolách bude popsáno základní nastavení programu. Budou popsány jednotlivé uživatelské prvky i způsob jejich implementace. Kapitola 3.4.1 popisuje samotné připojení k headsetu, nastavení jeho parametrů (při vytváření objektu třídy *Headset*). V kapitole 3.4.2 je popsáno jakým způsobem lze vybrat pouze konkrétní signály,

které chce uživatel zobrazit v režimu zobrazení více signálů. A poslední podkapitola 3.4.3 se věnuje možnosti tvorby a správy štítků, které později slouží pro vyznačování úseků v grafech.

3.4.1 Nastavení připojení k headsetu

Nastavení připojení k headsetu je možné v levé části s panely, konkrétně na panelu *Connection*. Na obrázku 3.3 je náhled aplikace, s tímto panelem a vstupními prvky, které musí uživatel zadat.



Obrázek 3.3: Nastavení připojení k headsetu.

Nyní popíšeme jednotlivé položky, které je potřeba zadat při vytváření objektu třídy *Headset*, který slouží k připojení a komunikaci se samotným headsetem:

- *Connection type* – Udává jakým způsobem se budeme připojovat k headsetu. Uživatel má možnost vybrat mezi třemi způsoby. První je **Headset** – tím se nastaví přímé připojení k samotnému headsetu, bez nutnosti využít další aplikace. Tato možnost zajistí připojení pomocí metody *EE_EngineConnect*. Druhou volbou je **EmoComposer** – toto nastavení slouží pro vývoj s rozhraním *Emotiv SDKLite*, které nezahrnuje samotný headset. Aplikační rozhraní tedy komunikuje s *EmoComposer*, což je *EmoEngine* emulátor. Nezískává tak data ze skutečného headsetu. Při této možnosti se využívá metoda *EE_EngineRemoteConnect*. Poslední možností je **EmotivControlPanel** – toto nastavení slouží pro připojení k aplikaci *Emotiv Control Panel*, která může sloužit jako proxy pro skutečný headset, integrovaný *EmoEngine* nebo *EmoComposer*. Jako v

předchozím případě se pro tuto možnost využije metoda *EE_EngineRemoteConnect* [11].

- *Host address* – Jde o IP adresu ve formátu IPv4. Toto nastavení slouží k určení adresy v síti, kde je spuštěn *EmoEngine*. Je tedy patrné, že se využije při nastavení typu spojení na **EmoComposer**, nebo **EmotivControlPanel**. Ve výchozím nastavení je tato hodnota nastavena na adresu *127.0.0.1*. Tato adresa je přiřazena *loopback* rozhraní, což je virtuální rozhraní samotného počítače.
- *Host port* – Jde o port počítače, který identifikuje společně s IP adresou konkrétní stroj a službu, v našem případě buď **EmoComposer**, nebo **EmotivControlPanel**. Výchozí nastavení je pro **EmoComposer** port *1726* a pro **EmotivControlPanel** port *3008*. Stejně jako u adresy i v tomto případě se nastavení využije pro tyto dva typy připojení.
- *Buffer Size in Seconds* – Nastaví velikost bufferu pro ukládání dat z headsetu. Jde o vnitřní buffer headsetu a velikost ovlivňuje jak často je nutné obsluhovat událost oznamující, že jsou připravena nová data.
- *User ID* – Nastavuje identifikátor uživatele. K hardwarovému přijímači může být připojeno více bezdrátových headsetů, proto je nutné pomocí tohoto nastavení rozlišit, o kterého uživatele jde.

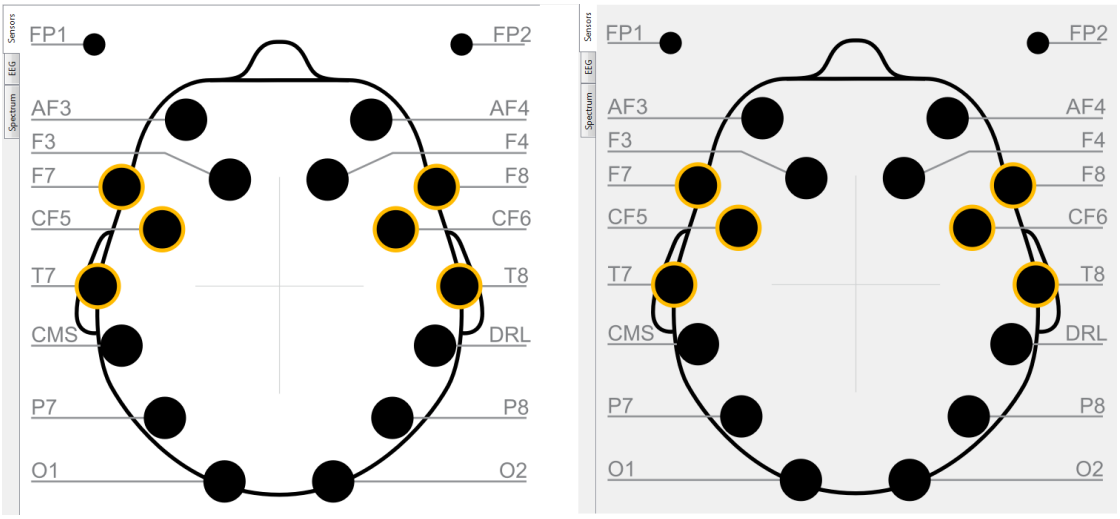
Dále se na tomto panelu nachází tlačítko *Start / Stop*, pomocí kterého lze spustit a poté zastavit zachytávání dat. Poslední dvě tlačítka *Safe Buffered Data* a *Load Data* slouží k uložení a načtení dat ze souboru, popis jejich funkce a formátu dat je uveden v podkapitole 3.9.

3.4.2 Výběr konkrétních signálů

Výběr konkrétních signálů, které chce uživatel zobrazit v grafech má smysl, pouze v režimu hromadného zobrazení signálů. Tento režim lze najít v pravé skupině panelů, konkrétně na panelu *EEG*. Nastavení výběru konkrétních signálů se pak provádí na panelu *Sensors*, který se také nachází v pravé skupině panelů. Vybírání konkrétních signálů je ukázáno na obrázku 3.4. Na obrázku jsou zobrazeny dvě situace – v levé části je aktivní vybírání signálů a v pravé neaktivní, kdy uživatel nemůže vybírat senzory kliknutím. Vybírání signálu je neaktivní, pokud je spuštěno zachytávání. Na obou obrázcích jsou vybrány senzory *F7*, *F8*, *FC5*, *FC6*, *T7* a *T8*, pro uživatele je to zobrazeno žlutým ohraničením daného senzoru. Ostatní senzory jsou neaktivní, tedy není kolem nich vykresleno žádné barevné ohraničení.

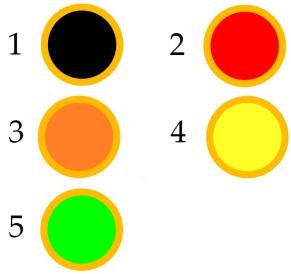
Na obrázku je zobrazeno celkem osmnáct senzorů, všechny však není možné vybrat. Jak bylo popsáno v tabulce 2.3 datových kanálů je celkem čtrnáct. Senzory *FP1*, *FP2* jsou lokalizační a senzory *CMS*, *DRL* jsou referenční. V grafech je tedy možné zobrazit čtrnáct kanálů, které lze vybrat. Po zapnutí programu jsou vybrány všechny senzory.

Po spuštění zachytávání a tím aktivování připojení k headsetu mohou senzory měnit svoji barvu, podle kvality přijímaného signálu. Na obrázku 3.5 jsou zobrazeny všechny



Obrázek 3.4: Výběr konkrétních signálů.

možné stavy (rozlišené barvami), ve kterých se senzory mohou nacházet.



Obrázek 3.5: Síla signálu jednotlivých senzorů.

Význam jednotlivých barev, vzhledem ke stavu úrovni signálu je následující:

1. Žádný signál – headset může být vypnutý, nebo jsou senzory špatně umístěny.
2. Velmi slabý signál.
3. Slabý signál.
4. Dobrý signál.
5. Plný signál.

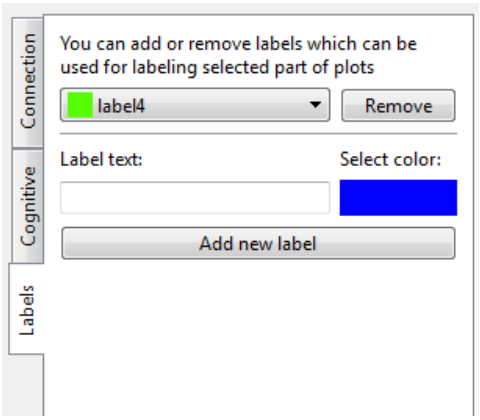
Pro kvalitní příjem signálu by měly být všechny senzory zelené. Nejprve je třeba správně zaměřit referenční senzory *CMS* a *DRL* a poté i ostatní. Senzory musí být dostatečně navlhčeny pomocí fyziologického roztoku. Poté je s nimi nutné na hlavě pohybovat, dokud není v programu ukázána plný síla signálu.

Jak bylo zmíněno v kapitole 3.3 o zobrazení tohoto obrázku se senzory se stará třída *myScene*. Tato třída dědí z vestavěné třídy frameworku *Qt* pro zobrazení povrchu pro grafické položky – *QGraphicsScene*. Obsahuje dvě důležité metody, které jsou *drawSensors* pro samotné vykreslení obrazové předlohy a *mouseReleaseEvent*, což je metoda obsluhující kliknutí do scény od uživatele. Dále tato třída obsahuje signál *sensorClicked*, pomocí kterého komunikuje s hlavním objektem třídy *SignalDockWidget*. Signál je zaslán při každém kliknutí na senzor a jako parametry obsahuje ID signálu a zda je aktivní či ne. V hlavním objektu, který se poté stará o vytváření grafů, je tento signál obsloužen a je uložena informace o tom, které senzory jsou aktivní. Tato informace je uložena ve struktuře *QList* v podobě pole čísel, které vyjadřují ID jednotlivých senzorů.

Předloha pro vykreslení senzorů je obrázek ve formátu SVG, který musí být v pevně stanoveném formátu. Pro vykreslení samotných senzorů je nejdůležitější část označená pomocí *id="LayerSensors"*. Každý objekt v této části je typu *circle* s vlastnostmi *id*, *fill*, *cx*, *cy*, *r* a senzory, které může uživatel vybírat mají navíc vlastnosti *stroke-width* a *channel-id*, který koresponduje s identifikátorem využitým pro identifikaci konkrétního kanálu při získávání dat z headsetu. Pro zpracování obrázku v tomto formátu se využívá modelu DOM. Konkrétně je to pak objekt třídy *QDomDocument*, který nabízí metody pro zpracování SVG obrázku, který se ukládá ve formátu XML.

3.4.3 Správa štítků

Důležitou částí programu je možnost vyznačovat v grafech určitý úsek dat. K tomuto účelu slouží tzv. štítkování. Než uživatel může využívat štítky, musí je nejprve vytvořit. Toto je možné provádět na panelu *Labels* v pravé skupině panelů. Zde má uživatel k dispozici jednoduché rozhraní pro správu štítků. Rozhraní je zobrazeno na obrázku 3.6.



Obrázek 3.6: Správa štítků.

V rozbalovacím seznamu jsou zobrazeny všechny uživatelovy štítky s barevnou ikonou a popisem. Po vybrání konkrétního štítku je možné pomocí tlačítka *Remove* tento štítek odstranit. Pro vytvoření nového štítku je nutné zadat jeho název a vybrat barvu. Vybrání barvy se provádí stiskem na barevný obdélník vedle políčka pro zadání textu.

Výchozí barva po zapnutí programu je nastavena na modrou. Pro uložení štítku do seznamu je nutné stisknout tlačítko *Add new label*. Pokud uživatel zadá již existující název, štítek se duplicitně neuloží. Takto vytvořené štítky má uživatel k dispozici na výběr v rozbalovacím seznamu v pravé skupině panelu na panelu *EEG*, při vybrání zobrazení všech zachycených dat. Toto bude podrobněji popsáno v kapitole 3.8.

Tyto lístky, které si uživatel vytvoří je nutné mít k dispozici i po zavření a opětovném spuštění programu. Proto nemohou být ukládány jen do dynamických struktur za běhu programu, ale musí být zároveň uloženy permanentně. K tomuto uchování dat slouží dvě metody v třídě *SignalDockWidget*, kterými jsou *safeLabels* – pro ukládání do souboru a *loadLabels* – pro načítání ze souboru. Metoda *loadLabels* se volá pouze v konstruktoru objektu třídy *SignalDockWidget* a metoda *safeLabels* je volána při každém smazání nebo přidání lístku. Obě metody ukládají data potřebné k vytvoření příslušných štítků do souboru s názvem *"labels.txt"*, který je uložen v adresáři s programem ve formátu:

`index_poradi_v_rozbalovacim_seznamu;nazev;id_barvy`

kdy každý řádek představuje jeden štítek. Při načítání se kontroluje formát souboru a pokud nějaký záznam (řádek v souboru) není v požadovaném formátu, štítek se neuloží. I při načítání štítků ze souboru se kontroluje duplicita.

3.5 Hromadné zobrazení v reálném čase

V těchto podkapitolách budou ukázány možnosti zobrazení dat v reálném čase, kdy má uživatel možnost sledovat několik grafů. Každý graf vykresluje data z příslušného senzoru. Kolik bude mít uživatel zobrazených grafů záleží na nastavení senzorů, které bylo popsáno v kapitole 3.4.2.

Toto hromadné zobrazení dat je dostupné v pravé části s panely, konkrétně na panelu *EEG*. Data je možné zobrazovat ve dvou režimech. Prvním z nich je zobrazení v časové oblasti, kdy se zobrazují surová data, jak jsou přijímána z headsetu. Podrobněji bude rozebráno v následující podkapitole 3.5.1. Druhým režimem je zobrazení ve frekvenční oblasti. Zde se již data zpracovávají a provádí se Fourierova transformace, opět bude podrobněji popsáno v podkapitole 3.5.2. Na tomto panelu je také možnost přepnout se do třetího režimu, kterým je zobrazení všech zachycených dat. Tento režim však neslouží pro zobrazení dat v reálném čase a bude podrobněji popsán v podkapitole 3.7.

Jak bylo popsáno v podkapitole 3.2 data jsou ve všech režimech zobrazována pomocí stromové struktury. Konkrétně je k tomu využit grafický objekt třídy *QTreeWidget*, ve kterém jsou dynamicky tvořeny objekty třídy *QTreeWidgetItem*. Tyto dynamické objekty poté v sobě obsahují objekt pro vykreslení samotného grafu. Tato stromová struktura obsahuje dva sloupce. V prvním sloupci je vždy uvedeno ID senzoru a jeho název, v druhém je pak samotný graf. Jelikož může být v panelu zobrazeno současně až čtrnáct grafů, je využito rolování ve stromové struktuře, aby nebyly grafy příliš malé.

Grafy se zobrazí až po stisknutí tlačítka *Start / Stop*. Do té doby je stromová struktura prázdná. Tedy po začátku zachytávání se vytvoří jednotlivé objekty. Samotný proces od stisku tlačítka pro zahájení zachytávání až po začátek zasílání dat lze shrnout do čtyř bodů. Ukážeme to na volání metod hlavní třídy *SignalDockWidget* a popisu co dělají:

1. *createHeadset* – Tato metoda vytvoří objekt třídy Headset, s předem zadanými parametry jak bylo popsáno v podkapitole 3.4.1.

2. *prepareSignalTreeWidget* – Tato metoda na základě předaného parametru vytvoří objekty ve stromové struktuře. Jejich počet závisí na počtu aktivovaných senzorů.

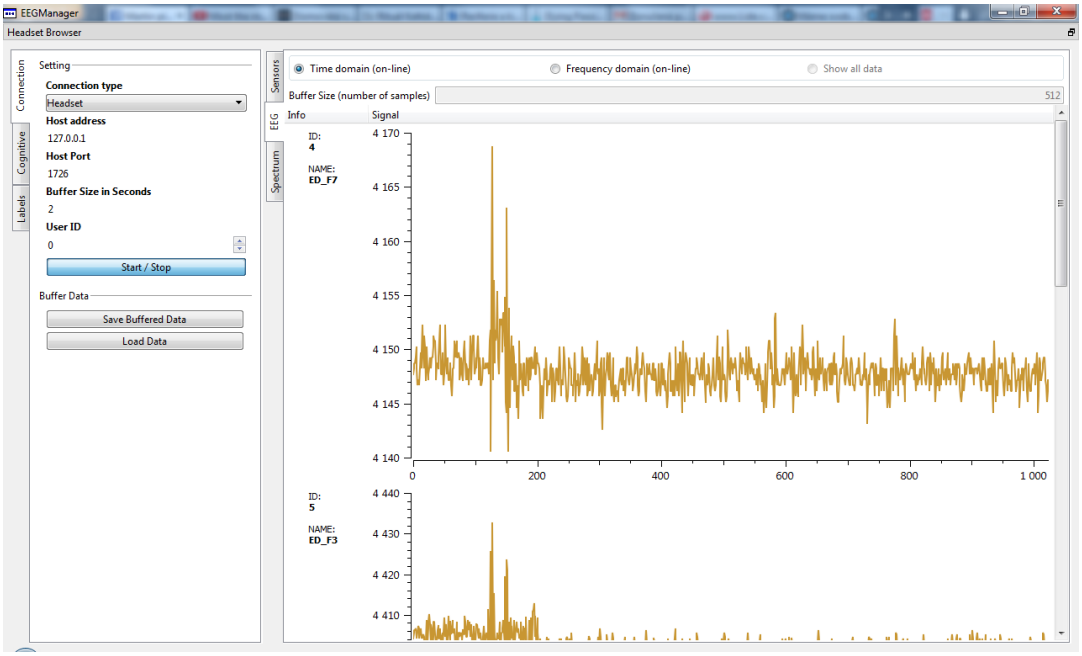
3. *connectPlotsToHeadset / connectPlotsToHeadsetFrequency* – zde se volá buď jedna, nebo druhá metoda, což záleží na vybraném režimu (zobrazení v časové oblasti – první metoda, nebo zobrazení ve frekvenční oblasti – druhá metoda). Obě metody pak připojí vytvořené grafické objekty ke zdroji dat.

4. *headset->start* – tato metoda přísluší objektu třídy *Headset*, který je vlákno. Touto metodou se spouští jeho činnost, tedy obsluha zachytávání dat z headsetu.

Po zahájení zachytávání může uživatel mezi dvěma režimy zobrazujícími data v reálném čase přepínat, aniž by musel zachytávání ručně zastavovat. Toto ruční zastavení se děje automaticky na pozadí programu. Tento přístup také přispívá ke snadnějšímu ovládání aplikace, bez nutnosti vykonávat zbytečné uživatelské operace navíc. Po každé změně režimu při spuštěném zachytávání se tedy automaticky zastaví aktuální zachytávání, smažou se aktuálně zachycená data, ve stromové struktuře se smažou a znovu vytvoří dané objekty a opětovně se spustí zachytávání. Mazání objektů ve stromové struktuře má na starosti metoda *clearPlotsEEG*, která jako parametr přijímá číslo aktuálního režimu. Opět jde o metodu hlavní třídy *SignalDockWidget*.

3.5.1 Zobrazení v časové oblasti

V této kapitole bude ukázáno, jak se v grafech zobrazují data v časové oblasti. Toto zobrazení tedy uživatel najde na panelu *EEG* v pravé skupině panelů, kde je nutné zatrhnout volbu *Time domain (on-line)*. Jak bylo popsáno v předchozích kapitolách nejprve si uživatel vybere senzory, které chce sledovat, poté spustí zachytávání tlačítkem *Start / Stop* a vytvoří se daný počet grafů. O zasílání dat jednotlivým grafům se stará objekt třídy *Headset*, který pomocí signálů komunikuje přímo s objekty třídy *SignalPlotSpectrum*. Tyto objekty signál obslouží a vykreslí samotná data. Vykreslují se tedy holá data, přímo jak je zasílá headset. Nová data se vykreslují okamžitě, hned jak jsou zaslána headsetem. Na obrázku 3.7 je ukázáno, jak vypadají takovéto grafy.

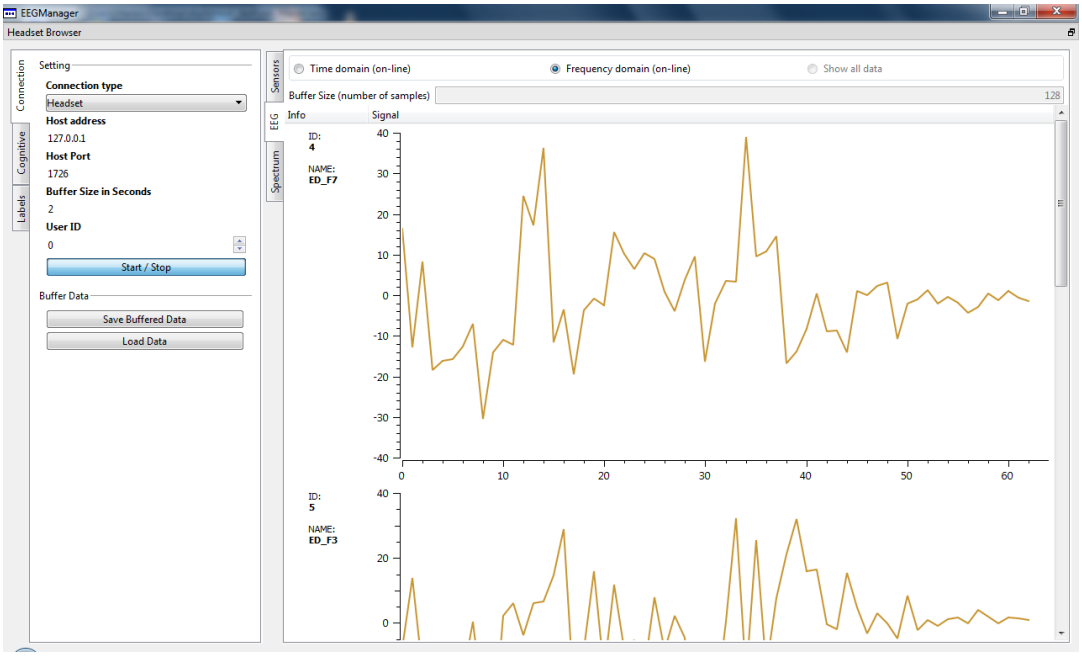


Obrázek 3.7: Zobrazení dat v režimu časové oblasti.

Z obrázku je patrné, že tlačítko *Start / Stop* je aktivováno a data jsou neustále přijímaná. Také je možné přepínat pouze mezi dvěma režimy (*Time domain* a *Frequency domain*). Na obrázku je možné vidět graf, pro který jsou zdrojem data ze senzoru *F7* a částečně graf s daty ze senzoru *F3*. Další grafy si uživatel může posouvat pomocí vertikálního posuvníku. Osa X označuje pořadí nasnímané hodnoty, má pevnou velikost 1024 vzorků. A osa Y pak značí velikost nasnímané hodnoty. Graf tedy zobrazuje posledních 1024 hodnot, které headset zaslal. Zatímco měřítko osy X je pevně dané, měřítko osy Y se přizpůsobuje aktuálně zobrazeným hodnotám. To samozřejmě přispívá k lepší čitelnosti grafu, než u pevně stanoveného měřítka.

3.5.2 Zobrazení frekvenční charakteristiky

Druhým režimem pro zobrazení dat v reálném čase je okno s frekvenční charakteristikou. Tento režim se opět nachází na panelu *EEG*, tentokrát je ovšem nutné zatrhnout volbu *Frequency domain (on-line)*. Opět se zobrazují pouze senzory, které si uživatel nastavil na panelu *Sensors*. Stejně jako při vykreslování grafů v časové oblasti i zde se po stisku tlačítka *Start / Stop* spustí zachytávání a vykreslí se daný počet grafů. Než uživatel ale spustí toto zachytávání musí zadat počet vzorků, tedy velikost bufferu, z kterých chce počítat Fourierovu transformaci. Toto je jediný nutný vstupní parametr, který musí zadat uživatel pro možnost výpočtu transformace a správného převedení signálu do frekvenční oblasti. Na obrázku 3.8 je ukázán graf, pro který jsou zdrojem data ze senzoru *F7* a částečně graf se zdrojovými daty ze senzoru *F3*, data jsou pomocí *FFT* převedena do frekvenční oblasti. Další grafy lze opět zobrazit pomocí svislého posuvníku.



Obrázek 3.8: Zobrazení dat v režimu frekvenční oblasti.

Stejně jako v režimu časové oblasti i zde při aktivovaném zachytávání je možné přepínat mezi těmito dvěma režimy. Nyní jsou v grafu na ose X zobrazeny jednotlivé frekvence v hertzích a na ose Y velikost amplitudy. Opět je měřítko osy X pevně dané a naopak měřítko osy Y se přizpůsobuje aktuálně zobrazených hodnotám. Hodnoty na frekvenční ose X jsou vždy rozmístěny od hodnoty nula až po hodnotu, která je vypočtena jako polovina čísla, které udává počet vzorků (parametr, který uživatel zadává do pole *Buffer Size (number of samples)*). Tato délka intervalu je dána Shannon-Kotelnikovým vzorkovacím teorémem, o kterém bylo zmíněno v podkapitole 2.4.1.

Pole se zadáním velikosti bufferu není pro uživatele nijak omezené. Pokud zadá jinou než číselnou hodnotu nastaví se délka na minimální hodnotu, což je šestnáct. Jak bylo zmíněno v podkapitole 2.4.1 algoritmus *FFT* musí mít délku signálu, která bude mocninou čísla dva. Jelikož program uživatele neomezuje pouze na výběr konkrétních délek, které by tomuto požadavku vyhovovaly, je potřeba ošetřit případ, kdy uživatel zadá délku, která nebude vyhovovat této podmínce. V programu se používá zaokrouhlení na nejbližší vyšší číslo, které je mocninou čísla dva. Je k tomu využit algoritmus 3.3, který vytvořil Sean Eron Anderson. Algoritmus pomocí bitové operace *OR* a bitového posunu zaokrouhlí 32 bitový *integer* na nejbližší vyšší mocninu čísla dva. Je k tomu zapotřebí pouze dvanáct operací. Algoritmus kopíruje nejvyšší nastavený bit do všech nižších a poté přičte hodnotu jedna. To způsobí, že všechny nižší bity jsou nastaveny na nulu a jeden bit před nejvyšším nastaveným bitem je nastaven na jedničku. První operace odečítá jedničku a je tu z důvodu, kdy by číslo již bylo zaokrouhlené. Po odečtení jedničky a provedení algoritmu se tedy vrátí na svoji původní hodnotu.

```
unsigned int v;
v--;
v |= v >> 1;
v |= v >> 2;
v |= v >> 4;
v |= v >> 8;
v |= v >> 16;
v++;
```

Výpis 3.3: Algoritmus zaokrouhlení čísla na mocninu čísla dva.

Je zřejmé, že oproti režimu zobrazení v časové oblasti, kdy byla data zasílaná objektem třídy *Headset* přímo grafickým objektům, zde musí být jistý mezikrok. Tento mezikrok provádí samotnou transformaci dat, do frekvenční oblasti. Proto v tomto režimu se spolu s daným počtem grafů vytvoří i stejný počet objektů třídy *dataTransform*. Originální data z headsetu jsou nejprve zaslána všem objektům této třídy, kde jsou zpracována a následně již zaslána přímo objektům třídy *SignalPlotSpectrum*. Tato komunikace probíhá pomocí signálů. Transformovaná data mají svůj vlastní buffer, což umožňuje uchovávat i původní originální data, která je možné uložit do souboru.

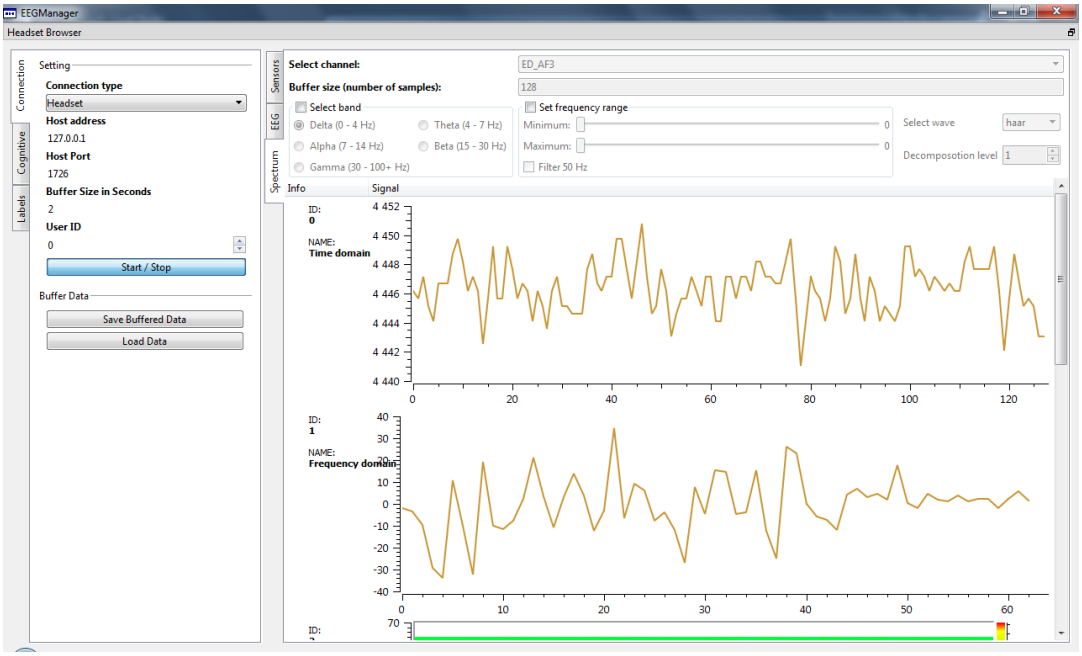
3.6 Zobrazení informací pro vybraný signál

Další možnost jak si uživatel může zobrazovat data v reálném čase je v režimu konkrétního signálu. Tuto možnost uživatel nalezne na panelu *Spectrum* v pravé skupině panelů. Struktura tohoto panelu je odlišná od struktury na panelu *EEG*. Opět je zde k dispozici více grafů, které uživatel může posunovat pomocí svislého posuvníku. Nyní však nevybírá které senzory budou zobrazeny na panelu *Sensors*, jak tomu bylo v případě hromadného zobrazení dat, ale zde vybere pouze jediný senzor pomocí volby *Select channel*.

Tento panel je rozdělen na dvě části. V první – té horní nalezne uživatel různé možnosti nastavení a ve druhé – té spodní se pak vykreslují samotné grafy. Grafy se zobrazí až po spuštění zachytávání pomocí tlačítka *Start / Stop*. Na obrázku 3.9 je zobrazen panel s dvěma grafy, konkrétně pro časovou a frekvenční oblast. Dále má uživatel k dispozici zobrazení spektrogramu a jednotlivých dekompozičních úrovní z vlnkové transformace. Všechny grafy mají jako zdroj svých dat pouze jediný senzor.

Než uživatel spustí zachytávání musí nastavit několik parametrů. Prvním z nich je, jak už bylo řečeno, výběr kanálu, který bude sloužit jako zdroj dat. Jelikož se v grafech využívá výpočtu Fourierovy a vlnkové transformace, která požaduje jako parametr velikost bufferu, je nutné zadat počet hodnot, z kterých se tyto transformace budou počítat. Pro nastavení tohoto parametru je zde položka *Buffer size (number of samples)*. Vstup opět není nikterak omezen a následně se v programu využívá stejného postupu jako v případě hromadného zobrazení dat ve frekvenční oblasti, tedy algoritmu pro zaokrouhlení čísla na nejbližší mocninu čísla dva. Samotná vlnková transformace potřebuje další dva parametry, podle kterých se řídí její výpočet. Prvním z nich je výběr mateřské vlnky. Pro tuto volbu je zde připraven rozbalovací seznam, kde má uživatel předdefinovány všechny vlnky, se kterými dokáže knihovna *wavelet1d* pracovat. Je označen textem *Select wave*. Jako

výchozí vlnka po zapnutí programu je vybrána *haarova* vlnka. A druhým parametrem, který funkce pro výpočet transformace vyžaduje, je počet úrovní rozkladu. Tato volba se nastavuje pomocí prvku typu *SpinBox* – zde pod označením *Decomposition level*. Tento prvek dovolí zadat uživateli pouze čísla v předem stanoveném rozsahu.



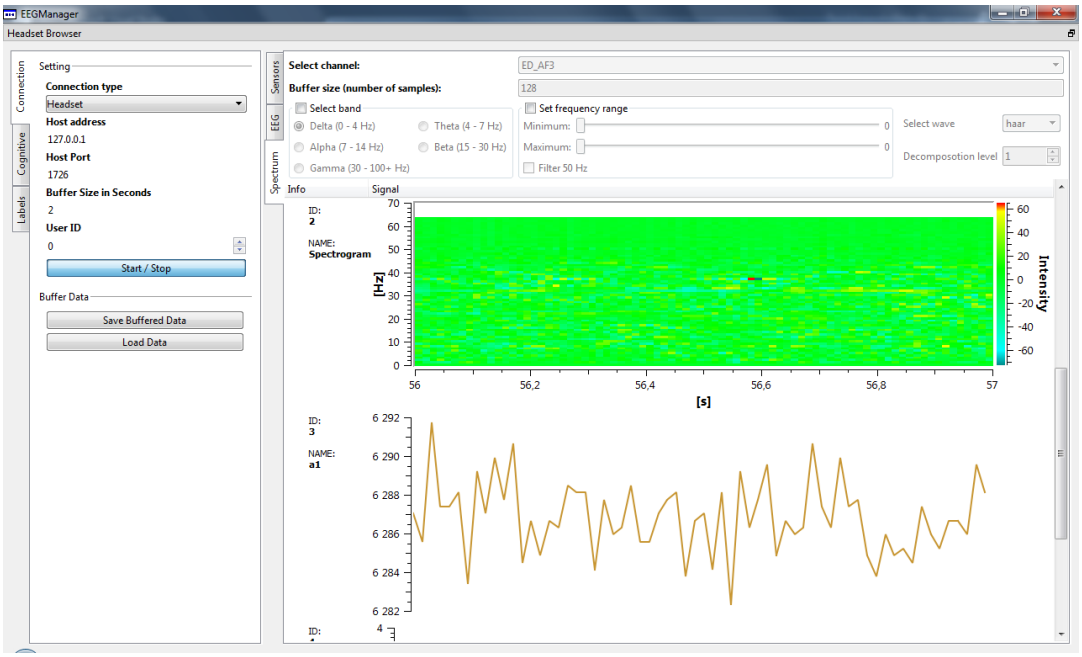
Obrázek 3.9: Zobrazení detailních grafů pro vybraný signál – časová a frekvenční oblast.

První dva grafy označené jako *Time domain* a *Frequency domain* jsou vizuálně stejné jako na panelu *EEG* v režimech časové a frekvenční oblasti. Chování obou os u grafů je tedy téměř stejné. Malý rozdíl je v délce a tedy počtu zobrazených hodnot u grafu *Time domain*. Zde je zobrazeno tolik hodnot, kolik uživatel nastaví jako velikost bufferu. Je to z důvodu, že zde zobrazení funguje trochu odlišně. Data v grafu pro časovou oblast zde nejsou přímo zasílána samotným headsetem, ale jsou nejprve zpracovávána. V tomto případě je nejprve provedena dopředná Fourierova transformace, kdy výstup se zobrazí v grafu pro frekvenční oblast, následně se provede zpětná Fourierova transformace a její výstup se zobrazí do grafu pro časovou oblast.

Tato změna souvisí s možnosti filtrace. K tomu slouží dvě skupiny prvků na tomto panelu v horní části s nastavením. Prvním z nich je volba filtrování podle konkrétních vln v signálu a druhá volba je možnost nastavit filtr dle uživatelem zvoleného rozsahu frekvencí. Možnost zapnout filtrování je vypnuta pokud není spuštěno zachytávání. Tato volba je tedy přístupná pouze při zobrazení dat v reálném čase. Uživatel si může zvolit současně pouze jednu skupinu filtru. Při aktivování filtru podle vln si uživatel může přepínat mezi konkrétními vlnami, které byly teoreticky popsány v podkapitole 2.2. U každé vlny je napsán její frekvenční rozsah, v grafu je však zobrazen rozsah o kousek vyšší aby uživatel neměl graficky oříznuté krajní frekvence. Naopak při výběru filtru

dle uživatelem zadaných frekvencí se v grafu zobrazí pouze zadaný rozsah. Zde může uživatel nastavit filtr od nuly až po polovinu hodnoty udávající velikost bufferu, což je maximum, jak bylo vysvětleno v podkapitole 3.5.2. Pokud je tedy aktivní jeden nebo druhý filtr, je tím ovlivněna také časová oblast grafu, jelikož do zpětné Fourierovy transformace vstupují filtrované hodnoty.

Na obrázku 3.10 je zobrazen graf pro spektrogram, označený jako *Spectrogram* a druhý graf, který zobrazuje jeden v výstupů vlnkové transformace, konkrétně aproximační úroveň a je označen jako *a1*.



Obrázek 3.10: Zobrazení detailních grafů pro vybraný signál – spektrogram a vlnková transformace.

Graf spektrogramu má tři rozměry. Na ose X je vynesen čas, na ose Y frekvence a osa Z, která je značena barvou nese informaci o amplitudě. V pravé oblasti grafu je pak legenda, která přiřazuje jednotlivým barvám hodnotu. Osy X i Y mají pevně dané měřítko. V případě osy Y je to opět polovina velikosti bufferu, jelikož se zde počítá Fourierova transformace. Osa X má pevné měřítko, které je dané počtem úseků, z kterých se spektrogram počítá. Poslední osa Z, zde barevná složka, mění svůj rozsah podle toho jaké je třeba zobrazit hodnoty amplitudy.

V tomto panelu se vykresluje několik typů grafů. Grafy pro časovou a frekvenční oblast spolu se spektrogramem jsou pevně dané. U grafů pro vlnkovou transformaci je však počet proměnný a záleží na tom, jakou úroveň dekompozice uživatel zvolil. Nejprve jsou tedy vytvořeny grafické objekty s grafy a poté se vytvoří objekty, které jim budou zajišťovat data. Vytvoří se jeden datový objekt třídy *dataTransform*, který je společný pro první dva grafy, dále se vytvoří jeden objekt třídy *dataSpectrum*, který slouží pro výpočet

spektrogramu. Jednotlivé grafy pro vlnkovou transformaci mají pak svůj společný datový objekt třídy *dataTransform*. Komunikace mezi headsetem, objekty které zpracovávají data a grafickými objekty je zajištěna pomocí signálů a slotů.

Aby práce se zobrazením dat v různých režimech byla pro uživatele co nejjednodušší a vyžadovala co nejméně zásahů, je možné přecházet mezi oběma panely *EEG* a *Spectrum*, aniž by bylo potřeba zastavovat zachytávání tlačítkem *Start / Stop*. Uživatel tedy může snadno přecházet mezi oběma panely a na panelu *EEG* mezi oběma režimy pro zobrazení dat v reálném čase. Každé takovéto přepnutí při zapnutém zachytávání dat způsobí, že se na pozadí programu zachytávání zastaví, vymažou se stávající zachycená data a opět spustí zachytávání v novém režimu. Efekt je tedy stejný, jako kdyby uživatel toto zachytávání sám zastavil, změnil režim a opět zachytávání spustil. Mazání objektů ve stromové struktuře pro tento panel zajišťuje metoda *clearPlotsSpectrum* třídy *SignalDockWidget*.

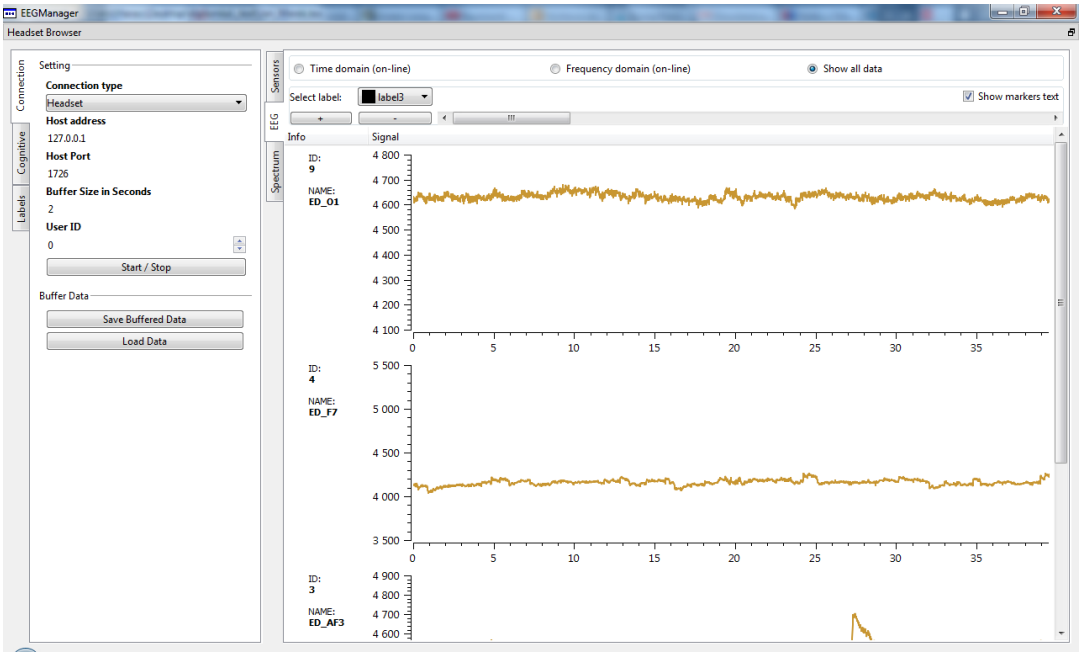
Na závěr zmíníme, že pokud uživatel na tomto panelu ručně zastaví zachytávání, v grafech budou zobrazena poslední zachycená data. I při změně panelu a vrácení se zpět na panel *Spectrum* zde budou tyto data zobrazena. Data budou smazána až opětovným spuštěním zachytávání, nebo ukončením programu.

3.7 Zobrazení všech zachycených dat

Poslední režim, který ještě nebyl popsán a ve kterém lze nahlížet na data je zobrazení všech zachycených dat. Jde o zobrazení neupravených dat v časové oblasti. Tuto možnost uživatel nalezne na panelu *EEG* v pravé skupině panelů. A zde konkrétně pod režimem *Show all data*. Tento režim však není určen pro zobrazování dat v reálném čase, proto při spuštění zachytávání je tato volba neaktivní. Do tohoto režimu se uživatel může dostat dvěma způsoby. Prvním z nich je zastavení zachytávání tlačítkem *Start / Stop*. Pokud je uživatel na panelu *EEG* v režimu *Time domain* a zastaví zachytávání aplikace se automaticky přepne do režimu *Show all data*. Pokud je uživatel v režimu *Frequency domain* a zastaví zachytávání aplikace zůstane ve stejném režimu. Nyní má uživatel možnost změnit režim na *Show all data* a zobrazit si tak všechny zachycená data v časové oblasti. Druhým způsobem jak zobrazit data v tomto režimu je využít možnosti načtení dat ze souboru. Takto načtená data se zobrazí právě v tomto panelu a tomto režimu. Více o této možnosti je popsáno v podkapitole 3.9.

V tomto režimu má uživatel opět k dispozici možnost ovlivnit, kolik bude zobrazeno grafů. Tedy stejně jako v ostatních dvou režimech může nastavit na panelu *Sensors*, které senzory mají být aktivní. Tuto změnu může provádět kdykoliv v tomto režimu a grafy se tak budou průběžně přidávat, nebo odstraňovat. Jednotlivé grafy jsou pak seřazeny pod sebou stejně jako v předchozích režimech a uživatel si je může posunovat pomocí svislého posuvníku. Při přepnutí do tohoto režimu se taky zobrazí panel s několika ovládacími prvky. Vrchní část panelu slouží pro ovládání grafů a spodní pak pro samotné zobrazení dat. Na obrázku 3.11 je uvedeno okno programu s několika grafy v režimu zobrazení všech dat.

Osa X má v těchto grafech délku, která odpovídá době zachytávání dat. Je to tedy časová osa udávající ve vteřinách jak dlouho bylo spuštěno zachytávání. Osa Y poté zobrazuje hodnotu amplitudy a stejně jako u grafů pro zobrazení dat v reálném čase se



Obrázek 3.11: Zobrazení všech zachycených dat.

její rozsah dynamicky přizpůsobuje zobrazeným datům, může mít tedy v každém grafu jiný rozsah, což u osy X neplatí.

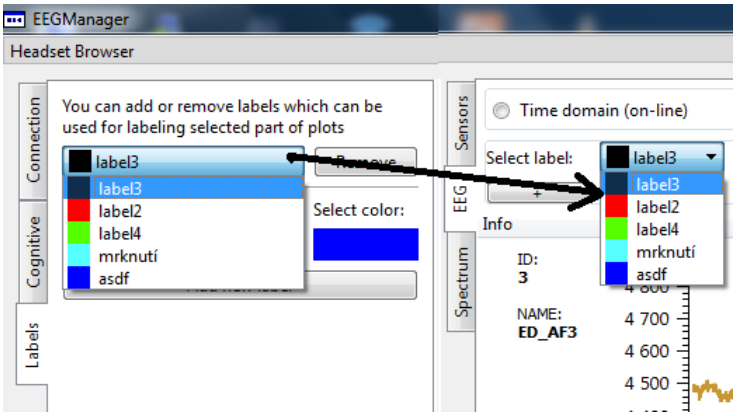
Zobrazení všech dat do grafu širokého jen dle šířky aplikace by bylo v tomto režimu velmi nepřehledné. EEG záznamy mohou být velmi dlouhé a obsahují mnoho hodnot. Proto je nutné aby měl uživatel možnost přibližovat různé úseky dat. Pro možnost přibližování a oddalování jsou k dispozici dvě tlačítka označená symboly "+"– pro přiblížení a "-"– pro oddálení. Vedle těchto tlačítek je pak horizontální posuvník, pomocí kterého je možné se v datech posouvat. Pomocí těchto ovládacích prvků se upravuje zobrazení všech zobrazených grafů najednou.

Pokud bylo zachyceno méně než deset hodnot, při zastavení zachytávání se nevykreslí žádné grafy. Jinak je pro každý graf vytvořen objekt třídy *SignalPlotAll*. Ten po vytvoření okamžitě zobrazí data z příslušného senzoru. Tento objekt také obsluhuje vstupy od uživatele. Grafy v tomto režimu nabízí možnost označování úseků dat. Tyto objekty tedy musí zpracovávat vstupy od uživatele, které poté pomocí signálů předávají k dalšímu obsloužení hlavnímu objektu třídy *SignalDockWidget*. Pro tuto interakci slouží zbytek ovládacích prvků v tomto panelu. Jsou jimi možnost výběru konkrétního štítku, který bude použit k označování dat. Tento výběr lze učinit pomocí rozbalovacího seznamu vedle nápisu *Select label*. Štítky v tomto seznamu jsou totožné s těmi, který uživatel vytvořil na panelu *Labels* v levé skupině panelů, jak bylo popsáno v podkapitole 3.4.3. Posledním ovládacím prvkem je zaškrtávací tlačítko *Show markers text*, pomocí kterého může uživatel zobrazit nápisy jednotlivých štítků přímo v grafech. Podrobněji budou možnosti a způsob označování rozebrány v podkapitole 3.8.

3.8 Označování vybraných částí grafu

Důležitá část aplikace je možnost vyznačovat úseky dat přímo v grafech. Toto označování musí být rychlé a jednoduše přístupné uživateli. Možnost takto označovat části grafu se nabízí v režimu zobrazení všech dat. Tedy poté co uživatel zastaví zachytávání, nebo si načte data ze souboru má možnost procházet data v grafech a dělat si k nim poznámky v podobě barevného označování daných úseků.

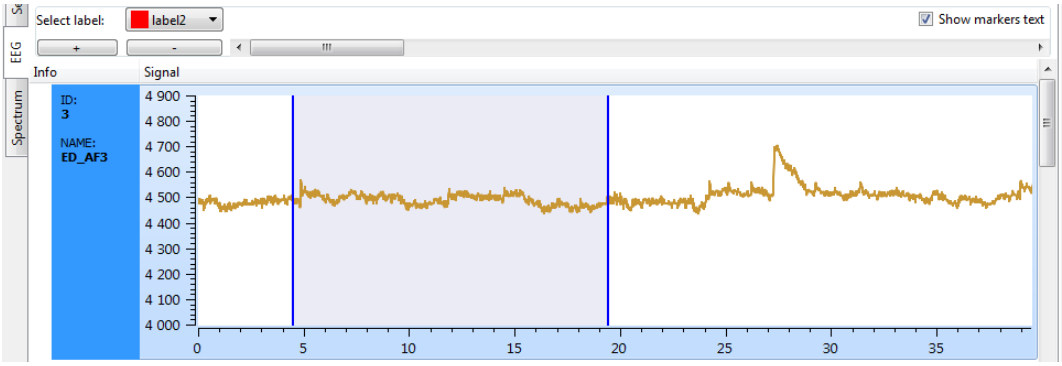
Nejprve tedy uživatel musí vytvořit štítky, které poté může vkládat do grafu. Tento postup byl detailně popsán v podkapitole 3.4.3. Poté má v režimu zobrazení všech zachycených grafů na výběr z rozbalovacího seznamu naprosto stejné štítky, které si vytvořil v části nastavení, jak je ukázáno na obrázku 3.12.



Obrázek 3.12: Vybírání štítků pro označování částí grafu.

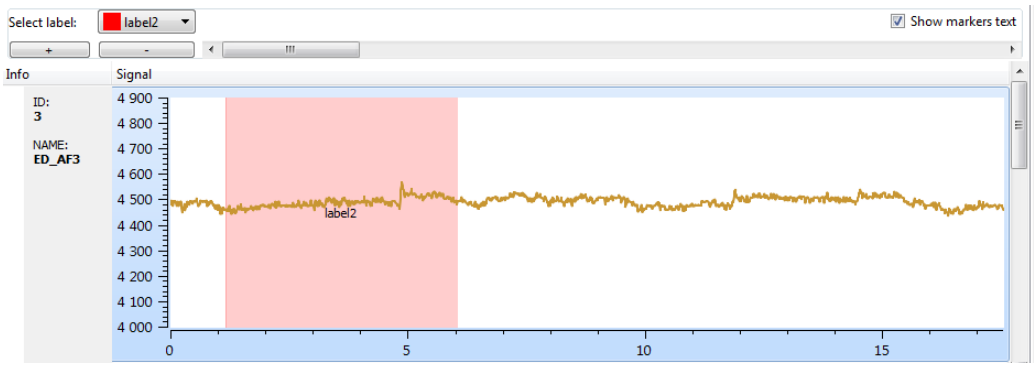
Pokud by uživatel žádný štítek nevytvořil a začal označovat úseky dat, tak se v grafech vytvoří štítek s výchozím nastavením barvy na černou a textem ”default”. Tedy program nutně nevyžaduje vytvoření štítků. Samotné barvy štítků jsou pak v grafech zobrazeny s průhledností 20%, aby nepřekrývali samotná data. Tato průhlednost má vliv pouze na pozadí, samotnou křivku neovlivní. Uživatel může vytvářet překrývající se štítky, kdy se pak průhlednost snižuje postupným překrýváním.

Samotná tvorba štítků se provádí pomocí tažení kurzoru horizontálním směrem přes oblast grafu. Tažení kurzorem se provádí při stisknutém levém tlačítku myši. Při tomto tažení se aktuálně vybíraná část zobrazí na daném grafu modrým zvýrazněním, kdy levý i pravý okraj je označen vertikální tučně modrou čárou. V místě, kde uživatel poprvé stiskne tlačítko se vytvoří pevný bod, od kterého se bude určovat vybraná oblast. Od tohoto bodu lze vybírat oblast nalevo, nebo napravo a druhý bod se bude měnit dle aktuální pozice kurzoru. Pokud uživatel vybere oblast, která je na ose menší než 0,01 vteřin oblast se neuloží. Toto je tedy minimální rozlišení, které uživatel musí vybrat. Ukázka vybírání části grafu tažením kurzoru je na obrázku 3.13.



Obrázek 3.13: Označování části grafu tažením kurzoru.

Poté co uživatel uvolní kurzor vybraná oblast změní barvu podle aktuálně vybraného štítku v rozbalovacím seznamu a uloží se do grafu. Zároveň se takto vybraná oblast vyznačí do všech zobrazených grafů. Na obrázku 3.14 je zobrazena takto vyznačená oblast pomocí štítku "label2"červené barvy. Dále je možné si všimnou, že na obrázku je přímo v grafu zobrazen popis štítku. To je způsobeno zatrhnutým tlačítkem *Show markers text*. Uživatel může pomocí tohoto tlačítka přímo v grafech zobrazovat názvy štítků. Pokud tlačítko nebude zatrhnuté, text se nebude zobrazovat.



Obrázek 3.14: Označená oblast v grafu.

Tento způsob označování je navržen tak, aby uživatel potřeboval k jeho provedení co nejméně akcí. Uživatel má samozřejmě také možnost takto vytvořené štítky mazat. Toho lze docílit stiskem pravého tlačítka kamkoliv ve vyznačené oblasti štítku. Pokud bude v oblasti více překrývajících se štítků, smažou se všechny. Opět je kladen důraz na co nejjednodušší obsluhu, která vyžaduje co nejméně akcí.

Při vytvoření štítku se uloží do dynamického pole záznam struktury *plotMarker*, který obsahuje informace potřebné ke grafickému zobrazení štítku. Toto dynamické pole je součástí objektu třídy *EEGBuffer*. Ukázka této struktury je na výpisu 3.4. Obsahuje položky pro uložení hraničních časů, identifikaci barvy a název štítku.

```
struct plotMarker
{
    double from;
    double to;
    uint flagColor;
    std::string flagName;
    plotMarker() : from(0.0), to(0.0), flagColor(0), flagName("") {}
};
```

Výpis 3.4: Struktura pro uložení informací o štítku.

Poté se vytvoří grafické objekty, které přímo reprezentují štítky v grafech. Jsou to objekty třídy *QwtPlotZoneItem* a vytvoří se objekt pro každý graf. Textový popis je pak objekt třídy *QwtPlotMarker*. Seznam těchto grafických objektů je poté uložen do dynamického pole, aby s nimi bylo možné jednoduše pracovat. Pro obsluhu událostí od uživatele nabízí knihovna *Qwt* třídu *QwtPlotPicker*. Každý graf má vytvořený dva objekty této třídy – jeden pro obsluhu vybrání štítku tažením kurzoru a druhý pro obsluhu stisku pravého tlačítka.

3.9 Ukládání a načítání dat

Důležitou součástí programu je možnost ukládat a také načítat zachycená data. Tuto volbu má uživatel na panelu *Connection* v levé skupině panelů. Slouží k tomu dvě tlačítka – *Save Buffered Data* a *Load Data*. Tyto tlačítka jsou neaktivní při zapnutém zachytávání. Vždy když uživatel spustí a následně zastaví zachytávání má možnost uložit všechna zachycená data, nezávisle na panelu a režimu ve kterém zachytávání spustil. Vždy se uloží všechna zachycená data v nezměněné podobě, přímo jak je zaslal headset.

Po stisknutí tlačítka pro uložení dat se otevře dialogové okno pro výběr složky, do které se všechny data uloží. Poté uživatel vybere složku a aplikace v ní vytvoří potřebné datové soubory. Ukládají se všechna zachycená data ze senzorů a další pomocné data. Těmi jsou informace o časovém průběhu a informace potřebné pro vytvoření štítků. Samotná data z headsetu jsou ukládána zvlášť pro každý kanál do samostatného souboru ve formátu čísel oddělených mezerou. Informace o časovém průběhu jsou uloženy zvlášť do souboru, kdy na prvním řádku je uvedena hlavička, která vysvětluje význam jednotlivých hodnot a na dalších řádcích pak samotná data. Posledním souborem je soubor s informacemi o zaznačených štítcích. Opět má formu, kdy na první řádek je uložena hlavička a na dalších řádcích pak samotná data. Jsou zde uložena všechny data ze struktury *plotMarker*, kdy na každém řádku je jeden záznam a jednotlivé položky ze struktury jsou odděleny mezerou.

Při stisknutí tlačítka pro načtení dat se také otevře dialogové okno pro výběr složky, ve které jsou uložena data ve stejném formátu jak bylo popsáno v předchozím odstavci při ukládání dat. Po vybrání složky jsou data načtena do vnitřní paměti programu a zobrazena v grafech na panelu *EEG* v režimu zobrazení všech dat.

Načítání i ukládání má na starost několik metod v různých třídách. Prvním krokem je zobrazení dialogového okna pro výběr složky, to mají na starosti metody třídy *SignalDockWidget*, tedy *on_pushButtonSave_clicked* a *on_pushButtonLoad_clicked* (první obsluhuje

tlačítko pro ukládání, druhá pro načítání dat). Druhý krok zajišťují metody třídy *Headset* – *saveBufferedData* a *loadBufferedData*. Tyto metody vytvoří názvy jednotlivých souborů pro samotné data a nakonec zavolají metody třídy *EEGBuffer*, které zajistí samotný zápis do souborů, nebo čtení ze souborů. Pro zápis jsou to metody *saveBuffer* – ukládá samotná data, *saveChunks* – ukládá informace o časování, *saveLabels* – ukládá informace o štítcích a pro načítání pak metody *loadBuffer* – načítá samotná data, *loadChunks* – načítá informace o časování, *loadLabels* – načítá informace o štítcích.

4 Testování

V této kapitole podrobíme aplikaci testování, které odhalí její limity. Bude se tedy jednat zejména o výkonnostní testy. Pokusíme se odhalit jaké jsou mezní limity pro různé parametry aplikace. Může to být například délka transformací, kdy je stále možné zobrazovat data v reálném čase bez zpoždění způsobené výpočtem.

Testování bude provedeno s headsetem od společnosti *Emotiv*, představeným v podkapitole 2.3. Testování bude provedeno na notebooku s těmito parametry:

- *procesor* – Intel Core 2 Duo T5870, 2000 MHz
- *paměť RAM* – 3 GB DDR2-667
- *grafická karta* – ATI Mobility Radeon HD 3470
- *operační systém* – Microsoft Windows 7 Professional

Testování bude probíhat po restartování systému, kdy bude spuštěn pouze program *EEGManager*.

Jako první nastavíme vstupní parametr určujícím délku pro výpočet transformací na hodnotu 128. To je z důvodu, že tento headset má maximální vzorkovací frekvenci 128 vzorků za vteřinu, je tedy schopen zobrazit signály v rozmezí 0–64 Hz. Při tomto nastavení musí být aplikace schopna pracovat bez znatelného zpomalení. Dalším parametrem který lze nastavovat je úroveň dekompozice pro vlnkovou transformaci. Maximální úroveň je však omezena vzhledem k velikosti vstupního bufferu. Minimální počet vzorků pro jednu úroveň je čtyři, to znamená, že při celkovém počtu vzorků 128 může být vypočítáno maximálně pět úrovní dekompozice. Nastavena bude tedy maximální možná úroveň dekompozice a vybrána Haarova vlnka.

Postupně bylo spuštěno zachytávání v režimech zobrazení dat v reálném čase, tedy časové a frekvenční zobrazení na panelu *EEG* (kde byly aktivovány všechny senzory) a detailní zobrazení pro jeden vybraný signál (zde AF3) na panelu *Spectrum*. V každém režimu bylo spuštěno zachytávání po dobu 100 vteřin a přes *Správce úloh systému Windows* kontrolováno minimální a maximální vytížení procesoru. To se ve všech režimech pohybovalo mezi 20–40%. Zároveň nebylo zpozorováno žádné zpomalení aplikace, které by uživateli bránilo v její obsluze.

Při dalším testu nastavíme velikosti vstupního bufferu na hodnotu 8192. Při takto nastaveném vstupním bufferu je možné zobrazit frekvenční rozsah 0–4096 Hz. To samozřejmě nemá význam u EEG signálů, aplikace však může sloužit i pro zpracování jiného typu signálu. Při spuštění v režimu zobrazení všech signálů ve frekvenční oblasti bylo vytížení procesoru mezi 45–55% a aplikace reagovala na rolování mezi grafy se znatelným zpožděním. Při takto velkém vstupním bufferu se v grafech zobrazuje velké množství dat a aplikace není schopna bez zpomalení dále obsluhovat interakce od uživatele.

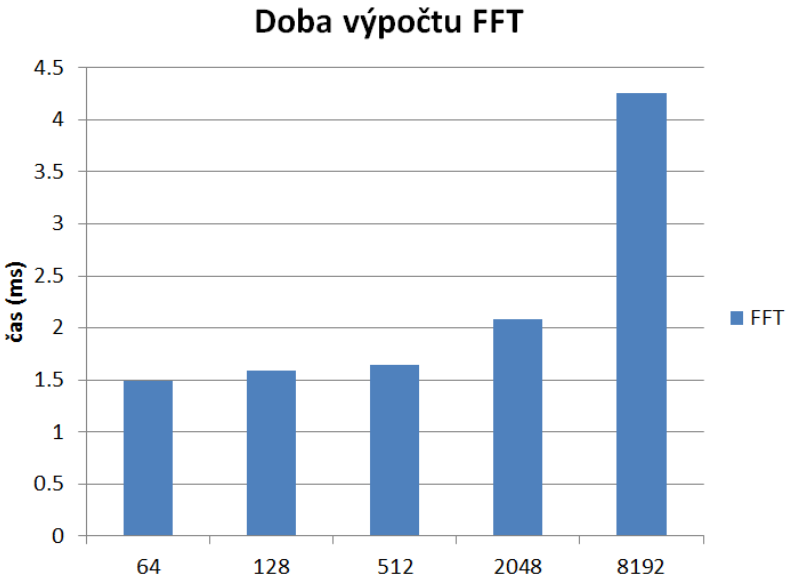
Nakonec ověříme jak program zvládá zpracovat dlouhé záznamy. Program bude spuštěn v režimu zobrazení všech grafů v reálném čase po dobu 30 minut. Poté bude zastaven,

čímž dojde k přepnutí do režimu zobrazení všech dat. Nakonec všechny data uložíme do souboru a poté zkusíme zpátky načíst.

Program při takto velkém množství dat zabírá v paměti RAM 107 MB. Pokud je v režimu zobrazení všech dat zvoleno malé přiblížení – to znamená jsou v grafech zobrazena téměř všechny data, bylo na testovaném počítači zpozorováno malé zpoždění na reakci posunu dat po časové oblasti a označování štítků. Pokud však uživatel zvolí vyšší přiblížení, tím tedy sníží aktuální počet zobrazených dat na displeji je reakce již plynulá. Při ukládání i načítání byla zpozorována prodleva několika vteřin (vždy maximálně 8 vteřin), než program opět reagoval na uživatelské vstupy. Uložená data měla velikost 63,4 MB.

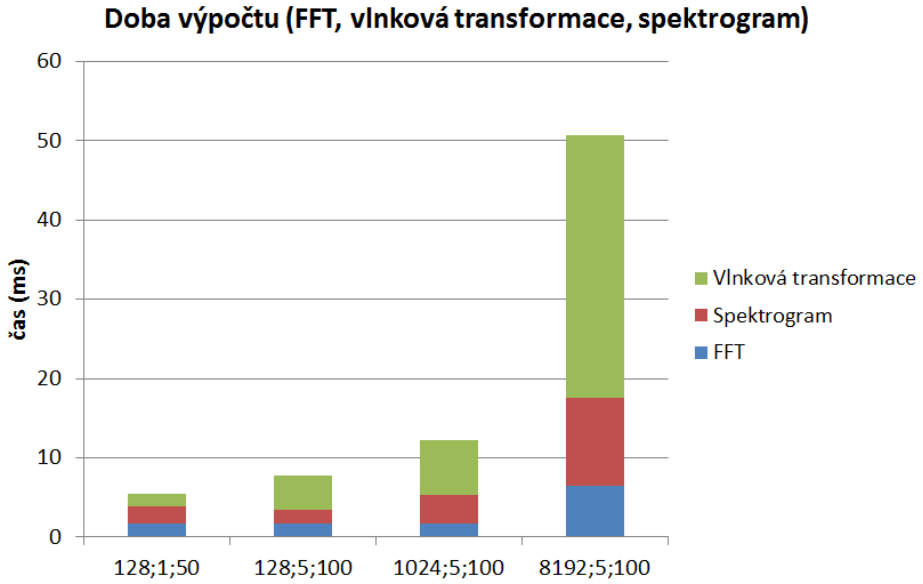
Testy, které byly provedeny byly hodnoceny uživatelem při práci s programem. Bylo posuzováno vytížení procesoru na základě pozorování pomocí programu *Správce úloh systému Windows*. V další části provedeme měření doby výpočtu jednotlivých transformací. Tento výpočet bude již přesný, jelikož bude využito měření doby výpočtu v programovém kódu aplikace. Využije se objektu třídy *QElapsedTimer*, který je součástí frameworku *Qt*. Bude se měřit doba výpočtu jednotlivé transformace (Fourierovy a vlnkové transformace, spektrogramu). Měření začíná při přijmutí signálu o dostupnosti nových dat od headsetu až po vyslání signálu grafickému objektu, který značí, že jsou dostupná data k zobrazení. Jde tedy o dobu přístupu k datům, jejich případné kopírování do pomocných struktur, dobu výpočtu samotné transformace a vyslání signálu.

První měření bylo provedeno při výpočtu Fourierovy transformace na panelu *EEG*. Testování bylo provedeno pro celkem pět různých velikostí bufferu (64, 128, 512, 2048, 8192 vzorků). Vždy bylo vybráno prvních dvacet hodnot měření a ty byly následně zprůměrovány. Doba výpočtu pro jednotlivé velikosti bufferu je v grafu 4.1.



Obrázek 4.1: Doba výpočtu Fourierovy transformace.

V druhém měření byla počítána doba pro výpočet všech transformací, které se zobrazují na panelu *Spectrum*. Zde se počítá několik transformací – dopředná a zpětná Fourierova transformace, vlnková transformace a výpočet spektrogramu. Opět bylo provedeno dvacet měření a výsledky zprůměrovány. Ve sloupcovém grafu jsou pak časy pro všechny transformace sečteny. Graf obsahuje čtyři sloupce, kdy každý sloupec představuje jiné nastavení programu. Sloupce jsou označeny třemi čísly oddělenými středníkem. První hodnota představuje velikost vstupního bufferu, druhá úroveň dekompozice vlnkové transformace a třetí šířku spektrogramu (tedy z kolika kroků Fourierovy transformace byl vytvořen).



Obrázek 4.2: Doba výpočtu transformací na panelu *Spectrum*.

Dále byl změřen interval, v jakém headset zasílá signál, kterým oznamuje zachycení nových dat – ten činí 101,48 ms. Z obou grafů je tedy patrné, že při žádném testovaném nastavení není výpočet prováděn déle, než je interval zasílání nových dat. Při velikosti vstupního bufferu 128 je rychlost výpočtu v obou panelech několikanásobně nižší, než doba za kterou jsou připravena nová data. Z tohoto důvodu můžeme říci, že výpočet transformací je dostatečně rychlý a neomezuje činnost programu.

5 Závěr

V práci byla popsána nejprve teorie z oblasti elektroencefalografie. Bylo nutné specifikovat tuto metodu a rozebrat jednotlivé její složky. Těmi jsou zejména samotný EEG signál a jeho vlastnosti. Dále jisté vzory, které se vyskytují v takovýchto signálech – vlny, u kterých bylo popsáno kdy se vyskytují, spolu s důležitým parametrem a to frekvenčním pásmem, které taktéž vymezuje jejich výskyt. Poslední složkou této oblasti byl popis způsobu zachytávání signálů. Zde byl také představen testovací headset od společnosti *Emotiv*, se kterým byla poté vyvíjena aplikace. Druhou částí teorie byl popis základních operací se signály. Byla popsána Fourierova a vlnková transformace, tvorba spektrogramu a možnost filtrování. Bylo popsáno k čemu jednotlivé operace slouží a jak fungují. Prakticky jsou pak použity v aplikaci při různých operacích.

Hlavním cílem bylo navrhnout a implementovat aplikaci umožňující práci s EEG signály. Nejprve bylo nutné rozvrhnout funkcionalitu programu, tedy jaké operace musí být pro uživatele dostupné. Program dokáže analyzovat signál pomocí výše zmíněných operací. Tuto analýzu umí zpracovávat a zobrazovat v reálném čase. Dále bylo důležité uživateli umožnit pracovat s dlouhými zachycenými záznamy. Samozřejmostí je možnost záznamy ukládat a poté znovu načíst do programu a zobrazit. V těchto dlouhých záznamech (EEG záznamy mohou být i několikahodinové) je nutné se velmi snadno orientovat. Touto orientací je myšleno zejména snadno nalézt v záznamu důležité úseky dat. V aplikaci je proto zahrnuta funkce pro zvýrazňování vybraných úseků dat. Takovýchto úseků může být v záznamu mnoho, proto aplikace nabízí uživatelsky velmi jednoduchou a rychlou možnost označování těchto úseků.

Při návrhu aplikace bylo nutné oddělit práci s daty a uživatelské rozhraní. V každé části bylo nutné vyřešit jednotlivé operace, které budou nutné pro celkové fungování aplikace. V datové části to bylo zejména získávání dat z headsetu, jejich ukládání do vnitřních struktur a operace nad nimi. Byly vybrány jednotlivé knihovny, které dané operace efektivně provádí. V části s uživatelským rozhraním poté bylo potřeba navrhnout pro uživatele jednoduché ovládání a snadný přístup ke všem funkcím. Opět zde byly vybraný grafické knihovny jednak pro celkovou tvorbu rozhraní a poté pro samotné vykreslování grafů.

Na závěr byla aplikace výkonnostně otestována, kdy se prověřila její dostatečná výkonnost pro všechny operace, které nabízí v rámci práce s reálnými daty. V testech byly změřeny doby výpočtu jednotlivých transformací a ukázalo se, že jsou několikanásobně nižší, než je interval zasílání nových dat od headsetu. V práci nejsou zahrnuty další pokročilé metody pro zpracování EEG signálů, o které by bylo možné aplikaci dále rozšířit. Mohou to být například *klasifikační metody*, *analýza hlavních komponent*, *výpočet příznaků*.

Aplikace byla vytvořena pro operační systém *Microsoft Windows* a při jejím vývoji testována s headsetem od společnosti *Emotiv*.

6 Literatura

[1] BALEK, Bronislav. Elektrické biosignály lidského těla měřené ISESem. V: *Souhrnný sborník Veletrhu nápadů učitelů fyziky* [online]. Olomouc, 2001 [cit. 2014-02-06]. Dostupné z: http://vnuf.cz/sbornik/prispevky/16-01-Balek.html

[2] VOJTĚCH, Zdeněk. Využití elektroencefalografie v současnosti. *New EU MAGAZINE of MEDICINE* [online]. 2009, 1-2, s. 43-64 [cit. 2014-02-06]. Dostupné z: http://www.neumm.cz/public/img/neumm_09_1-2/pdf/vojtech_encefalografie.pdf

[3] EMOTIV. *Emotiv | EEG System | Electroencephalography* [online]. [cit.2014-02-13]. Dostupné z: http://www.emotiv.com

[4] EPOC Specifications. *Emotiv | EEG System | Electroencephalography* [online]. [cit. 2014-02-13]. Dostupné z: https://www.emotiv.com/epoc/download_specs.php

[5] TŮMA, Jiří. *Zpracování signálů získaných z mechanických systémů užitím FFT*. Praha: Sdělovací technika, 1997, 174 s. ISBN 80-901-9361-7.

[6] JAKSCH, Ivan. *Technická diagnostika*. Liberec, 2011. Učební text. Technická univerzita v Liberci.

[7] ANISIMOVA, Elena, Jan BEDNÁŘ a Petr PÁTA. Zpracování obrazu pomocí vlnkové transformace. *Elektrorevue* [online]. 2013, č. 4, s. 238-246 [cit. 2014-02-20]. Dostupné z: http://www.elektrorevue.cz/cz/clanky/zpracovani-signalu/0/zpracovani-obrazu-pomoci-vlnkove-transformace--image-processing-using-the-wavelet-transform-/

[8] RAJMIC, Pavel a Zdeněk PRŮŠA. Podrobná studie algoritmu pro výpočet waveletové transformace v diskrétním čase. *Elektrorevue* [online]. 2010, č. 6 [cit. 2014-02-20]. Dostupné z: http://www.elektrorevue.cz/cz/clanky/zpracovani-signalu/15/podrobna-studie-algoritmu-pro-vypocet-waveletove-transformace-v-diskretnim-case/

[9] *10/20 Positioning System: Manual*. Hong Kong: Trans Canial Technologies, 2012.

[10] MALMIVUO, Jaakko, Robert PLONSEY. *Bioelectromagnetism: principles and applications of bioelectric and biomagnetic fields*. New York: Oxford University Press, 1995, ISBN 01-950-5823-2.

[11] EMOTIV. *Emotiv Software Development Kit: User Manual*.

[12] BEDNAŘÍK, Josef, Zdeněk AMBLER a Evžen RŮŽIČKA. *Klinická neurologie*. Vyd. 1. Praha: Triton, 2010. ISBN 978-807-3873-899.

[13] VOJTĚCH, Zdeněk. *EEG v epileptologii dospělých*. 1. vyd. Praha: Grada, 2005. ISBN 80-247-0690-3.

[14] MITRA, Partha a Hemant BOKIL. *Observed brain dynamics*. New York: Oxford University Press, 2008. ISBN 01-951-7808-4.

[15] ROZMAN, Jiří. *Elektronické přístroje v lékařství*. Vyd. 1. Praha: Academia, 2006. ISBN 80-200-1308-3.

[16] ŠMÍD, Radislav.*Úvod do vlnkové transformace*. Praha, 2001. Učební text. České vysoké učení technické v Praze.

[17] JAN, Jiří. *Číslicová filtrace, analýza a restaurace signálů*. Vyd. 2. Brno: VUTIUM, 2002, 427 s. ISBN 80-214-1558-4.

[18] KOZUMPLÍK, Jiří; KOLÁŘ, Radim; JAN, Jiří. *Číslicové zpracování a analýza signálů : počítačová cvičení*. Brno : VUT, 2003.

[19] 1-D Decimated Wavelet Transforms: MATLAB & Simulink. MATHWORKS. *MathWorks: MATLAB and Simulink for Technical Computing* [online]. [cit. 2014-03-10]. Dostupné z: `http://www.mathworks.com/help/wavelet/ug/one-dimensional-discrete-wavelet-analysis.html`

[20] Spectrogram using short-time Fourier transform: MATLAB spectrogram. MATHWORKS. *MathWorks: MATLAB and Simulink for Technical Computing* [online]. [cit. 2014-04-12]. Dostupné z: `http://www.mathworks.com/help/signal/ref/spectrogram.html`